UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
DIRECCIÓN DE FORMACIÓN POSTGRADUADA

SISTEMA PARALELO DE APOYO AL PROCESO DE EXPLORACIÓN DEL NÍQUEL

TESIS PRESENTADA EN OPCIÓN AL

TÍTULO DE MÁSTER EN INFORMÁTICA APLICADA

Autor: *Ing.* Asnay Guirola González

Tutor: *Lic.* Dannier Trinchet Almaguer

Ciudad de La Habana, diciembre de 2010

"La ciencia es la verdadera escuela moral; ella enseña al hombre el amor y el respeto a la verdad, sin el cual toda esperanza es quimérica"
Berthelot

DECLARACIÓN JURADA DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste se firma la presente a los _____ días del mes de _____ del año _____.

_____

Ing. Asnay Guirola González
Firma del autor

_____

Lic. Dannier Trinchet Almaguer
Firma del tutor

Agradecimientos

- *A todos aquellos que de una forma u otra han hecho posible el desarrollo de la tesis y me han apoyado en todo momento, no tengo palabras para agradecerles su colaboración sin la cual no fuera posible el desarrollo de este trabajo.*
- *Un agradecimiento muy especial a mi tutor por su asesoramiento científico, por su disposición permanente e incondicional en aclarar mis dudas, por su paciencia, esmero y ayuda.*
- *A mis padres, por su constante apoyo, comprensión, por su ejemplo de superación incansable; por lo que soy y seré gracias a ustedes, son los mejores padres del mundo.*
- *A mis familiares y amigos, por siempre estar presentes en mi vida.*

Dedicatoria

"A mis padres y hermano."

Resumen

Realizar exploraciones con el objetivo de identificar posibles zonas ricas en minerales es prioridad de la Industria Cubana del Níquel, para ello se emplean varias técnicas, una de ellas consiste en tomar muestras del suelo a diferentes profundidades y separadas a una determinada distancia una de otras. A estas excavaciones se les llama pozo y la información resultante de este proceso se almacena y emplea para lograr un adecuado modelado de los yacimientos lateríticos. Con este objetivo el Centro de Investigaciones del Níquel en Moa ha propuesto un modelo matemático que persigue alcanzar una alta representatividad de la realidad del yacimiento. Estos modelos Markovianos específicamente presentan el inconveniente que su aplicación requiere de altos requerimientos computacionales de tiempo y memoria, debido a su alta complejidad temporal y espacial, que obstaculizan su incorporación al proceso de Optimización de Redes de Exploración del Níquel.

Se propone en la presente investigación un sistema paralelo con el propósito de disminuir el uso de memoria y tiempo necesario para el cálculo numérico del modelo Markoviano y su incorporación a la optimización de redes de exploración y explotación de yacimientos lateríticos. Se muestra un análisis teórico del algoritmo empleado y se implementa utilizando la librería de Interfaz de Paso de Mensaje (MPI) para su ejecución sobre un clúster Beowulf, empleando tres modos de almacenamiento para el modelo obtenido. Además de realizar los análisis correspondientes a los resultados analíticos y empíricos evidenciando que el sistema es capaz de disminuir la cantidad de memoria y tiempo necesario para el modelado y es viable su incorporación al proceso industrial.

Palabras claves: Computación Paralela, Computación de Alta Productividad, Modelación de Yacimientos Lateríticos, Modelo Markoviano.

Abstract

Scans in order to identify potential mineral-rich areas is a priority of the Cuban nickel industry, for which several techniques are employed, one of them consists on taking soil samples at different depths and at a certain distance apart from each other. These excavations are called well and the information resulting from this process is stored and used to achieve an adequate modeling of lateritic deposits. To this end the Nickel Research Center at Moa has proposed a mathematical model that aims to achieve a high representation of the reality of the reservoir. These models, specifically the Markov model, have the disadvantage that their application requires high computational requirements of time and memory, due to its high temporal and spatial complexity that impede their incorporation into the process of Nickel Exploration Network optimization.

It is proposed in this study a parallel system in order to reduce the memory and time usage needed for the numerical calculation of the Markov model and its incorporation into the exploration and exploitation network optimization of lateritic deposits. It is shown a theoretical analysis of the used algorithm and it is implemented using the library of Message Passing Interface (MPI) for its execution on a Beowulf cluster, using three storage ways for the model obtained. In addition to performing tests to the analytical and empirical results showing that the system is able to reduce the amount of memory and time required for modeling and can be incorporated into industrial process.

Keywords: Parallel Computing, High Productivity Computing, Lateritic Reservoir Modeling, Markov Model.

Índice general

Introducción

La minería no llegó a constituir una actividad de primera importancia para la economía cubana hasta después de la segunda mitad del siglo XX. El mayor auge en la producción minera en la etapa anterior a 1959 se alcanzó en los períodos de confrontaciones bélicas cuando se incentivaba la producción minera en Cuba vinculadas a las guerras mundiales y a la de Corea. Existía desconocimiento del potencial minero del territorio nacional y una participación casi nula de inversionistas cubanos.

Al triunfar la Revolución en 1959 se toma la decisión de establecer un programa encaminado a precisar y desarrollar el potencial minero del país. En 1961 se crea el Ministerio de Industrias, existiendo sólo dos geólogos cubanos, por lo que se requería un importante proceso de preparación de condiciones en ese sentido. A partir de la creación del Servicio Geológico Nacional se comenzó un trabajo sistemático encaminado a precisar las características geológicas del país y revelar la presencia de yacimientos de minerales, lo cual trajo como resultado el desarrollo de una fuerte base de reservas minerales para la industria del níquel con un alto grado de confiabilidad.

Cuba cuenta con las principales reservas de níquel del mundo, lo que ha permitido desarrollar una industria que está representada por tres plantas, donde los yacimientos lateríticos proporcionan la materia prima fundamental de la Industria Cubana del Níquel y es muy conocida su complejidad en cuanto a la composición química y mineralógica de los materiales que en ellos yacen.

La Industria Cubana del Níquel persigue el aumento de la eficiencia y eficacia durante su proceso de producción y con este la productividad, para ello ha elevado el ritmo de extracción y utilización de la materia prima mineral [8]. La eficiencia y eficacia de este proceso de exploración de minerales es dependiente de la calidad con que se realice el muestreo del yacimiento [8, 37], el cual puede ser mejorado actualmente y de esta forma aumentar los niveles productivos [18, 26].

Los yacimientos presentan una alta complejidad en cuanto a su composición química

y mineralógica [17, 31] lo que provoca que la exploración y evaluación geólogo-económica de yacimientos lateríticos sea difícil y los principales esfuerzos estén dirigidos a lograr la mayor exactitud en el pronóstico de reservas minerales útiles [18, 26].

El Centro de Investigaciones del Níquel (CEINNIQ) de Moa, tiene como uno de sus principales objetivos mejorar el proceso de Optimización de Redes de Exploración del Níquel (ORENI), lo cual ha fomentado investigaciones que han brindado un grupo importante de resultados como un modelo matemático que persigue alcanzar una mayor representatividad de la realidad del yacimiento a modelar [23].

El paso inicial consiste en modelar matemáticamente el yacimiento, a partir de lo cual se optimiza el muestreo mediante un segundo modelo y posteriormente se optimizan las aproximaciones a utilizar en el pronóstico de los elementos químicos. La presente investigación está orientada a la obtención del primero de estos modelos, el cual es de tipo probabilístico, en específico Markoviano, basándose en la existencia de clases de materiales patrones y está encargado de modelar matemáticamente el yacimiento.

En estos tipos de yacimientos, se asume que el conjunto de estas clases contienen todas las variables presentes en ellos y que representan el espacio muestral. Cada clase se asume como uno de los estados en un proceso de Markov que persigue describir la dependencia existencial entre ellas, y donde las transiciones entre cada par de estados quedan condicionadas por otras cuatro variables que describen geoespacialmente el yacimiento [24].

Su matriz de transferencia se asocia a una matriz de particularidades espaciales que puede verse como un hipercubo de probabilidades condicionales 6-dimensional, que para su construcción se tienen que comparar entre sí todas las muestras del yacimiento a modelar, pues se necesita determinar para cada combinación de estos parámetros la aparición condicionada de las clases patrones. Este proceso es costoso computacionalmente, debido principalmente a que se genera una gran cantidad de iteraciones sobre el registro de muestras tomadas en el yacimiento, agravado por la necesidad de calcular, para cada una de estas, la influencia que tienen entre sí las clases observadas en cada punto de muestreo, que aunque es constante, no depende del tamaño del problema pero si provocan un costo de cómputo importante.

Se han propuesto algunos algoritmos secuenciales para su obtención, y aunque el desempeño mejora, son de complejidad $O(n^2)$ respecto al tamaño del origen de información. En algunos casos 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 2,66GHz y una memoria RAM de 2GB. En otras soluciones se realizaron optimizaciones basadas fundamentalmente en el desenrollado de ciclos y los tiempos fueron mejorados, para ese mismo tamaño de la entrada y medios de cómputo, hasta 7 horas 43 minutos y 16 segundos [27][30].

Para tamaños del problema mayores y más cercanos a los que generalmente se modelan en los yacimientos lateríticos, el cual aunque depende de la complejidad del yacimiento, 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.

El modelo genera un hipercubo de 6 dimensiones, esto hace que la complejidad del problema no se limite a la cantidad de cómputo necesario para su obtención, sino además, al espacio necesario para su búsqueda y almacenamiento, teniendo que emplearse memoria secundaria como alternativa ante la necesidad de memoria principal. Suponiendo que las 6 variables que describen el modelo tienen como promedio una cardinalidad igual a 30, entonces el modelado exigiría memoria para unos 729 millones de elementos que, si son considerados como reales en una arquitectura de 32 bit, emplearían 2.72 GB para su almacenamiento.

Esta situación origina el siguiente **problema científico**: ¿Cómo modelar yacimientos lateríticos con un bajo consumo de memoria y tiempo para que se facilite su incorporación al proceso de Optimización de Redes de Exploración del Níquel?

El principal **aporte práctico** esperado con la realización de este trabajo es brindar un sistema paralelo para la exploración de yacimientos lateríticos mediante la implementación del modelo propuesto por Peña en el 2007 [24] con la característica de poder ser ejecutado en un cluster de computadoras y logre disminuir la cantidad de memoria y tiempo necesario para el modelado y sea viable su incorporación al proceso industrial. Donde el **objeto de estudio** es la Computación Paralela aplicada a la exploración de yacimientos lateríticos.

Existen varios paradigmas de implementación paralela desde el punto de vista de la arquitectura del hardware con que se cuenta o la infraestructura de software a emplear dando lugar al uso de grandes mainframes o supercomputadoras. En el caso de nuestro país no es común encontrar arquitecturas que brinden un desempeño elevado, sin embargo, mediante la programación paralela aplicada a la exploración de yacimientos de níquel utilizando la modelación propuesta por Peña para ser ejecutado en un cluster Beowulf será el **campo de acción**.

De aquí que esta investigación tenga como **objetivo general**: Desarrollar un sistema paralelo para modelar yacimientos lateríticos, que valide experimentalmente su viabilidad para ser introducido en el proceso de Exploración de la Industria Cubana del Níquel.

Para cumplir el objetivo general de la investigación se trazaron los **objetivos específicos** que se exponen a continuación:

1. Analizar y diseñar un sistema paralelo para el modelado de yacimientos lateríticos siguiendo el modelo de Peña [23].
2. Implementar un sistema paralelo para su ejecución en un cluster de computadoras.
3. Evaluar el sistema experimentalmente.

Fue identificada como **Hipótesis investigativa** para la presente investigación, si se emplean técnicas de Computación de Altas Prestaciones para la exploración de yacimientos lateríticos a partir del modelado de los mismos y en correspondencia con las necesidades y posibilidades de cómputo del Centro de Investigaciones del Níquel, entonces es posible disminuir el tiempo y la memoria requerida por el modelado.

Variables:

- Tiempo de ejecución.
- Memoria necesaria para el modelado.

Los Métodos de **Investigación Científica** a utilizar son los siguientes:

- El *análisis documental* permite realizar un análisis de la información existente siendo utilizado durante el proceso de revisión bibliográfica que se realiza desde un enfoque histórico-lógico.
- *El análisis y la síntesis*. El *análisis* es un procedimiento mental mediante el cual un todo complejo se descompone en sus diversas partes y cualidades, permite la división mental del todo en sus múltiples relaciones y componentes. La *síntesis* establece mentalmente la unión entre las partes previamente analizadas y posibilita descubrir las relaciones esenciales y características generales entre ellas, se produce sobre la base de los resultados obtenidos previamente en el análisis.

- La *modelación*, método mediante el cual se crean abstracciones con vistas a explicar la realidad.

La tesis está estructurada en Resumen, Introducción, tres Capítulos como cuerpo fundamental de la tesis, Conclusiones, Recomendaciones, Referencias Bibliográficas, Anexos y Lista de Símbolos y Abreviaturas. Los capítulos corresponden a los diferentes aspectos temáticos:

En el Capítulo 1 *Fundamentación Teórica* se realiza un acercamiento al proceso de exploración del níquel y dentro de este al modelado de yacimientos lateríticos. Se presenta el objeto de estudio de la investigación y el modelo matemático propuesto por el Centro de Investigación del Níquel. Se exponen los trabajos que lo han empleado persiguiendo su introducción en el proceso industrial. Además son abordadas las principales clasificaciones y arquitecturas de los entornos que persiguen altas prestaciones, algunas consideraciones sobre el diseño y análisis de algoritmos paralelos.

En el Capítulo 2 *Descripción y Análisis de la Solución Propuesta* se describe la solución al problema planteado, partiendo de un acercamiento secuencial y luego se presenta un algoritmo paralelo del cual se muestra el análisis teórico de sus principales parámetros que permiten evaluar su rendimiento. Además de abordar las librerías necesarias para el diseño e implementación del sistema.

En el Capítulo 3 *Análisis de los Resultados* se presentan los resultados obtenidos por el sistema en su ejecución sobre un cluster Beowulf. Se realizan análisis comparativos en cuanto a los resultados obtenidos de forma teórica y empírica, concluyendo mediante una discusión sobre el cumplimiento de la hipótesis de investigación planteada.

1

Fundamentación Teórica

En el presente capítulo se realiza un estudio de algunos aspectos fundamentales del proceso de exploración del níquel y del modelado de yacimientos lateríticos. Se presenta el objeto de estudio de la investigación y el modelo matemático propuesto por el Centro de Investigación del Níquel. Se exponen los trabajos que lo han empleado persiguiendo su introducción en el proceso industrial. Además son abordadas las principales clasificaciones y arquitecturas de los entornos que persiguen altas prestaciones y algunas consideraciones sobre el diseño y análisis de algoritmo paralelos.

1.1 Proceso de producción del níquel

En el proceso de producción del Níquel y el modelado de yacimientos lateríticos la información geológica que debe ser almacenada y consultada en las empresas de exploración y extracción del níquel es sumamente amplia caracterizada por su volumen y por la diversidad de su contenido. Conjuntamente con las bases de datos de los pozos de perforación, obligatorias para el cálculo adecuado de las reservas, se precisa de información de carácter geográfico y múltiples bases de carácter específico, no geoquímico, que almacenen el resultado de pruebas de laboratorio a muestras seleccionadas, tales como los análisis granulométricos, mineralógicos, petrográficos, de sedimentación, entre otras [33].

La organización adecuada de tal volumen de información es fundamental para su empleo posterior en la modelación de los yacimientos, lo cual permite en gran medida la rapidez con que puedan tomarse determinadas decisiones referentes a la exploración y extracción del mineral.

La Industria Cubana del Níquel desde hace varias décadas ha promovido la investigación en el modelado [14][20][23], exploración[18, 37, 39] y evaluación geólogo-económica de yacimientos lateríticos [6, 8, 29].

La modelación matemática de un yacimiento es fundamental para la reconstrucción del cuadro geológico tridimensional del mismo, a partir de datos aislados obtenidos a través del muestreo. Este proceso combina la imaginación con las formulaciones matemáticas idóneas, de modo que con la información disponible, que generalmente es insuficiente debido al costo de su obtención, pueda lograrse un modelo adecuado,[32] el cual es de vital importancia para cumplir el objetivo principal de optimizar el proceso metalúrgico en general que permita lograr una industria puntera a nivel mundial tanto en niveles de producción como en calidad y costo [13].

Durante el proceso productivo, la Optimización de Redes de Exploración del Níquel (ORENI) es un hito importante para lograr disminuir los costos de exploración de yacimientos lateríticos útiles [8, 37]. Esto implica encontrar la topología óptima de la red en cuanto a cantidad de pozos, contorno que forma la unión de estos, densidad de pozos por zonas y todo lo referente a la distribución espacial de las perforaciones incluyendo su profundidad y la dirección de las conexiones entre los nodos. En general, las redes de exploración pueden verse como mallas rectangulares, donde los pozos son distribuidos uniformemente en cuadrículas [38].

Según *Peña y Legra* en el 2005 [26] plantean que el proceso ORENI esta regido por el Muestreo, el Modelado, la Optimización y por último el Pronóstico. El Muestreo es

el acopio de la información del yacimiento mediante una Red de Exploración la cual consiste en mallas regulares de perforaciones en el área del yacimiento para tomar muestras espaciadas hasta alcanzar la roca madre que le dio origen al yacimiento, en la figura 1.1 se muestra este proceso.

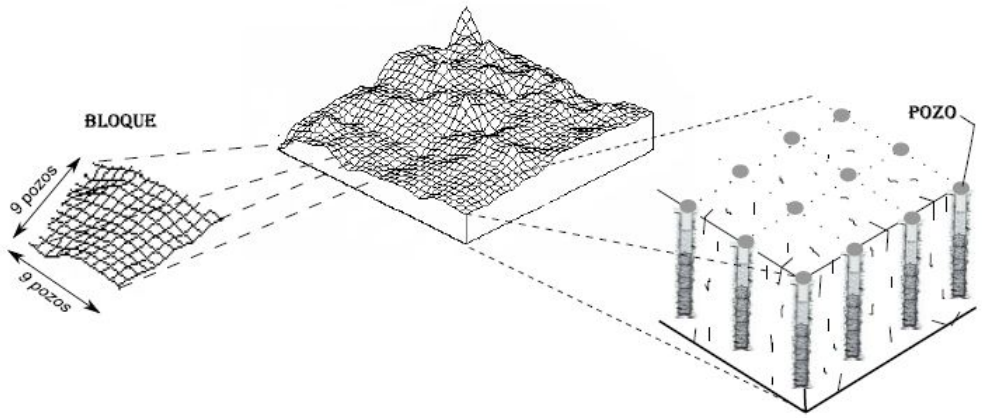

Figura 1.1: Red de Exploración

La Determinación de Clase tiene como objetivo clasificar el yacimiento a partir de clases patrones que se obtienen de los resultados de las variables medidas obtenidas en el muestreo y esta clasificación es empleada para modelar matemáticamente el yacimiento dando lugar al Modelado. Este modelo es Optimizado utilizando otros modelos matemáticos que tienen en cuenta características de tipo ambientales, legales y económicas, lo que conlleva que también quede optimizada la Red de Exploración y pueda determinarse las aproximaciones para cada zona del yacimiento en la nueva red permitiendo realizar Pronósticos de reservas minerales útiles.

Este proceso presentado en la figura 1.2 es cíclico y se repite hasta lograr un adecuado pronóstico en consecuencia con una alta certeza de la presencia del mineral en el lugar donde se realiza la exploración o se observen parámetros que indiquen baja o escasa presencia y se decida desistir su búsqueda.

1.2 Modelo de Peña

La influencia del muestreo en la eficiencia de la cadena productiva se convirtió en un problema que motivó, en el Centro de Investigaciones del Níquel de Moa, la búsqueda de

Figura 1.2: Optimización de Redes de Exploración

una solución siguiendo un nuevo enfoque [24, 26], este unificó en un modelo matemático varios aspectos que no se habían tratado hasta entonces como un sistema.

En [24] Peña en el 2007 propone tres modelos para optimizar este proceso, el paso inicial consiste en modelar matemáticamente el yacimiento, a partir de lo cual se optimiza el muestreo mediante un segundo modelo y en el tercero se optimizan las aproximaciones a utilizar en el pronóstico de los elementos químicos.

El primero de estos modelos se construye a partir del muestreo realizado al yacimiento, específicamente de la información correspondiente a todas las muestras tomadas y almacenadas en una tabla que contiene para cada muestra, tres campos llaves (pozo, bloque y profundidad) y un campo con la clase patrón observada. Esto da lugar a un modelo de tipo probabilístico, en específico un Modelo Markoviano el cual se basa en la existencia de tipos o clases de materiales patrones en los yacimientos, se asume que el conjunto de estas clases contiene todas las variables presentes en ellos y que representa el espacio muestral. Cada clase se asume como uno de los estados en un proceso de Markov, teniendo como base una Matriz de Probabilidades de Transición (MPT). Formalmente una matriz de transición M se define como:

$$M = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \diagdown & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{pmatrix} \tag{1.1}$$

Donde n es la cantidad de estados en los cuales puede estar el proceso a modelar (en nuestro caso la cantidad de clases), y p_{ij} la probabilidad de que el sistema pase del estado i al j. M cumple que:

$$\forall i, j \in [1, n] : 0 \leq p_{ij} \leq 1 \text{ y } \forall i \in [1, n] : \sum_{j=1}^{n} p_{ij} = 1$$

En la propuesta de Peña del 2007 [27][1] esta MPT se asocia a una matriz de particularidades espaciales y se obtiene:

$$X(t, r) = \sum_{i=1}^{n} \pi(i) * \rho(t, r) \tag{1.2}$$

Las matrices de transición $\pi(i)$ son matrices de probabilidades condicionales que en la ecuación 1.2 están asociadas a características geoespaciales $\rho(t, r)$, por lo que para toda combinación r de valores posibles de las variables que describen la dependencia entre las características observadas para cada par de puntos de muestreo en el momento t se tiene una matriz de transferencia elemental de $m * m$, donde m es la cantidad de clases que caracterizan a los yacimientos lateríticos según [24]. Las variables que se consideraron en r son: dirección horizontal (α_H), vertical (α_V) y teniendo en cuenta que no hay simetría en la dependencia entre las características de los yacimientos se consideró también la profundidad (H) y la distancia entre los puntos de muestreos (P); las que unidas a la característica observada en cada uno de los dos puntos de muestreo determinan el hipercubo de probabilidades de transición que representa el modelo. En la figura 1.3 se muestra una abstracción del modelo equivalente a una matriz de 6 dimensiones, que para su construcción se tienen que comparar entre sí todas las muestras del yacimiento a modelar, pues se necesita determinar para cada combinación de estos parámetros la aparición condicionada de las clases patrones.

Este conteo es costoso computacionalmente debido fundamentalmente a que se genera una gran cantidad de iteraciones sobre la tabla de las muestras tomadas en el yacimiento

[1]Propuesta de Clases Patrones en Yacimientos Lateríticos Ferro-Niquelíferos

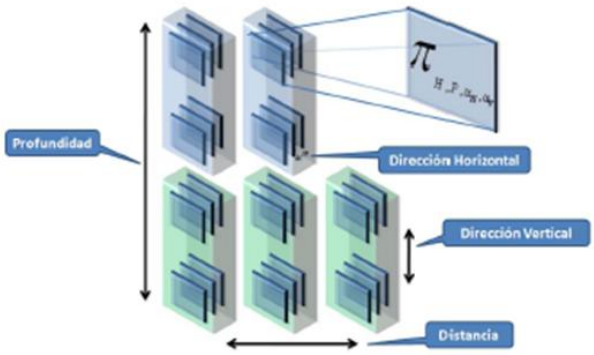

Figura 1.3: Abstracción del Modelo

a modelar y la representatividad del modelo estará dada por su precisión, la cual en algunos casos siendo baja es suficiente pero en otros una alta exactitud implica una alta complejidad.

Por otro lado, aunque la cantidad de muestras depende de la complejidad del yacimiento a modelar, en general las redes de exploración arrojan cientos de miles [7, 38], y en ocasiones pueden llegar al orden de los millones si se combinan, para un mismo yacimiento, los registros de múltiples muestreos. Este hecho fija el tamaño del problema a solucionar el cual es la cantidad de muestras arrojada por la red de exploración y tratadas en la etapa Determinación de Clases del proceso ORENI [26]. Los parámetros *tamaño del problema* y *representatividad del modelo* van en detrimento de la variable Tiempo y Memoria respectivamente; lo que provocan que la aplicación de este modelo usando una máquina convencional no sea viable en términos prácticos. En la siguiente sección se muestran algunos resultados de investigaciones que han tratado este problema.

1.3 Soluciones previas y sus principales limitaciones

Los primeros intentos de informatizar la minería del níquel en Cuba se remontan al año 1985, cuando se desarrolló el Sistema Níquel [10] cuyo objetivo principal era automatizar el manejo de la base de datos geoquímicas e informatizar las tareas de pronóstico, planificación y control de la minería en yacimientos lateríticos con el fin de hacer óptima la explotación de éstos.

Para que se tenga una idea de las escalas en que se ha trabajado hasta el momento

en los yacimientos lateríticos, en el año 2000 se extrajeron y transportaron al proceso metalúrgico alrededor de $3{,}5 \times 10^6$ T de mineral para extraer aproximadamente 30×10^3 T de níquel más cobalto [1].

Esta alta productividad conllevó al desarrollo de varias investigaciones por el Centro de Investigaciones del Níquel con el objetivo de optimizar el proceso ORENI, dentro del proyecto Software para la Optimización de Redes de Exploración, en este marco Peña en el 2007 [23] propone el modelo 1.2 visto anteriormente, para el cual se han desarrollado un grupo de herramientas computacionales para su obtención.

1.3.1 Solución secuencial según Peña

La solución propuesta por Peña en el 2007 [27] logra obtener el modelado pero con un alto costo de complejidad temporal ($O(n^2)$). Los experimentos fueron realizados con un tamaño del problema de 28511 registros, y la cardinalidad de las variables que determinan el modelo se fijaron en dirección horizontal $\alpha_H = 7$, vertical $\alpha_V = 17$, profundidad $H = 52$, distancia entre los puntos $P = 9$ y la cantidad de clases $m = 16$. En una PC dedicada con procesador Intel (R) Core (TM) 2 Duo CPU E6700 a una velocidad de 2,66 GHz y una memoria RAM de 2 GB. El tiempo necesario para el modelado fue de 17 horas, 10 minutos y 16 segundos, y el total de memoria requerida para el almacenamiento y obtención ascendió aproximadamente a 700 MB.

1.3.2 Solución paralela según Peña

Una implementación propuesta por Peña en el 2009 [25, 28] que emplea estaciones de trabajo conectadas mediante una LAN y para el mismo tamaño del problema y dimensiones del modelo, usando 5 estaciones de trabajo, el tiempo fue disminuido hasta 6 horas, 35 minutos.

Otras pruebas además, a partir de la información colectada en un yacimiento de mayor extensión, con un total de muestras de $n = 101343$ registros. Los tiempos haciendo uso de 4 estaciones de trabajo dedicadas, fueron estimados en aproximadamente 43 horas, siendo interrumpido el proceso debido a la no disponibilidad de las PCs.

1.3.3 Solución paralela según Trinchet

Un algoritmo propuesto por Trinchet en el 2010 [36] y su ejecución con el empleo de hasta 30 estaciones de trabajo conectadas mediante una LAN, para tamaños del problema de hasta 1024 mil y las mismas dimensiones del modelo dieron como resultado que el tiempo para el proceso de modelado se disminuyeran al igual que el consumo de memoria siendo esta la solución más óptima hasta la actualidad.

1.3.4 Principales limitaciones de las soluciones analizadas

Las principales limitaciones de las soluciones analizadas es que sus implementaciones no presentan un buen rendimiento desde el punto de vista de complejidad temporal y espacial.

En la solución secuencial las principales característica que dan al traste con un buen desempeño es el hecho de almacenar el modelo en memoria secundaria como alternativa ante la necesidad de memoria principal. Para cada Matriz de Probabilidades de Transición que conforma el modelo es necesario crear un fichero de texto que almacene la mencionada información, provocando que para una acción de actualización de un elemento, que conlleva una complejidad de $O(n^2)$, es necesario en primer lugar determinar en cuál de los ficheros de texto se encuentra, luego buscarlo en disco, cargarlo en memoria, modificarlo y finalmente actualizar el fichero.

La solución paralela se ve afectada en algunas ocasiones por el uso de tipos de datos pocos adecuados, como es el caso de algunos parámetros, identificación de los pozos y bloques de la red de exploración los cuales emplean cadena en lugar de valores numéricos, haciendo más lento todo el proceso de cálculo de otros parámetros que dependen directamente de estos.

Otras características que también afectaron esta solución están relacionadas con la arquitectura propia de la solución la que determinó una baja modularidad y el no establecimiento de capas bien definidas que permitieran una alta independencia entre la interfaz de usuario y el algoritmo, así como el no empleo de concurrencia que permitiera el monitoreo de la ejecución del algoritmo.

La solución paralela propuesta por Trinchet es la más acertada para realizar la implementación de un sistema que cumpla con los requisitos necesarios y viabilidad práctica para su incorporación al proceso industrial. La principal limitación está dada a que emplea estaciones de trabajo conectadas en una red no dedicada aprovechando sólo los recursos

disponibles en un momento determinado para obtener su solución. Y como se recomendó es prudente encontrar nuevas estructuras de datos que permitan reducir aun más el consumo de memoria.

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 los tiempos de ejecución resultan excesivamente elevados. Una forma de subsanar las limitaciones de las máquinas secuenciales es el procesamiento paralelo.

La Computación Paralela es una rama de la Ciencia de la Computación que estudia cómo resolver problemas haciendo uso de un conjunto de procesadores que son capaces de trabajar de forma cooperativa [12]. Se basa en el principio de que el proceso de solucionar un problema usualmente puede dividirse en pequeñas tareas que pueden ejecutarse simultáneamente y en coordinación. Lo cual permite disminuir el tiempo de ejecución

en la solución a problemas costosos computacionalmente y aumentar la dimensión de los problemas.

La Computación Paralela ha sido aplicada para resolver problemas científicos e ingenieriles complejos. Algunos ejemplos que pueden citarse son: El diseño de superficies para la aviación, circuitos de alta velocidad, estudio de fenómenos cuánticos, dinámica molecular, obtención de modelos estocásticos, simulaciones y estudios de la tierra, la atmósfera y el clima; entre otros [3, 4, 12].

Las arquitecturas para procesamiento paralelo han evolucionado, y en la actualidad las redes de computadoras constituyen una plataforma de cómputo paralelo muy utilizada por sus ventajas en términos de la relación costo/rendimiento. La noción de sistema distribuido como máquina paralela es común a las denominaciones redes de computadoras, redes de trabajo, clústeres, multi-clusteres y grid. En estos casos, se deben identificar las capacidades de procesamiento, interconexión, sincronización y escalabilidad [5].

La clasificación de sistemas paralelos según, los mecanismos global de control presentes, descrita por Flynn [11] se rige a partir de la ausencia o presencia de un control global que permite regular el flujo de instrucciones y datos de la plataforma paralela. Así es posible tener un flujo único ($S = single$) o múltiple ($M = multiple$) de instrucciones ejecutándose sobre un conjunto único ($S = single$) o sobre varios ($M = multiple$) conjuntos de datos.

La clasificación de Flynn incluye las siguientes categorías:

1. **Single Instruction stream, Single Data stream (SISD)** : Un flujo único de instrucciones se ejecuta sobre un único conjunto de datos. Los computadores SISD representan la mayoría de las máquinas seriales.

2. **Single Instruction stream, Multiple Data stream (SIMD)** : Un flujo único de instrucciones se ejecuta sobre diferentes conjuntos de datos. Los arreglos de procesadores están en esta categoría.

3. **Multiple Instruction stream, Single Data stream (MISD)** : Diferentes flujos de instrucciones se ejecutan sobre el mismo conjunto de datos. En la práctica es difícil encontrar máquinas reales que respondan a este modelo y esta estructura ha sido catalogada como impráctica por algunos arquitectos de computadora y actualmente no hay máquinas de este tipo, a pesar de que ciertas MIMD puedan ser usadas de esta forma.

4. **Multiple Instruction stream, Multiple Data stream (MIMD)** : Diferentes flujos de instrucciones se ejecutan sobre diferentes conjuntos de datos. Desde un

punto de vista teórico representa el modelo más versátil de los cuatro. La mayoría de los multiprocesadores y multi-computadoras pueden ser clasificados bajo esta categoría.

5. **Single program, multiple data (SPMD)** : Esta categoría no fue definida por Flynn, sin embargo es muy usada actualmente y puede ser considerada como un caso particular de ***MIMD***. Este modelo ocurre cuando cada procesador ejecuta una copia exacta del mismo programa, pero opera sobre datos diferentes.

Según la sincronía del reloj se pueden encontrar desde modelos donde existe un único reloj global que sincroniza todas las operaciones, iguales o distintas, en todos los procesadores dando lugar a un modelo **Sincrónico** hasta el caso contrario, en el que cada procesador tiene su propio reloj de forma tal que sea una clasificación **Asincrónica**, pasando por el caso intermedio, bastante frecuente, en el que sin permitir que un reloj global sincronice todas las operaciones, sí existen mecanismos que permiten sincronizaciones periódicas de los distintos elemento de proceso, ya sea a requerimiento del programador o de forma espontánea cada cierto período de tiempo dando lugar a una variante Híbrida entre las dos anteriores clasificaciones.

Otro parámetro de clasificación es la Interconexión, característica que rige el mecanismo mediante el cual los elementos pueden intercambiar información existiendo dos alternativas, el empleo de una **Memoria Compartida** o **Memoria Distribuida** empleando el paso de mensajes entre los distintos procesadores [22].

La organización de los procesadores o red se refiere a como se conectan o enlazan los procesadores o nodos en un computador paralelo. Entre los criterios que existen para evaluar los distintos diseños está el diámetro de la red el cual viene dado por la mayor distancia entre dos nodos. Mientras menor sea el diámetro menor será el tiempo de comunicación entre nodos arbitrarios. Otro criterio importante es el ancho de bisección de la red el cual esta definido por el menor número de enlaces que deben ser removidos para dividir la red por la mitad. Un ancho de bisección alto es preferible porque puede reducir el tiempo de comunicación cuando el movimiento de datos es crítico, ya que la información puede viajar por caminos alternos y así evitar o reducir la congestión entre ciertos nodos de la red. Igualmente un ancho de bisección alto hace el sistema más tolerante a fallas debido a que defectos en un nodo no hacen inoperable a todo el sistema.

Los principales tipos de organizaciones de redes de procesadores han venido condicionando hasta hace poco el modelo de programación asociado a la computadora paralela [4]; dando lugar a dos modelos, programación mediante variable compartida y paso de mensaje. Ambos son empleados en las diferentes topologías usadas para el diseño de redes

de interconexión en sistemas multiprocesadores [3, 40] las cuales tratan de permitir elevados desempeños en función de su costo y escalabilidad. Los principales diseños de redes de interconexión son:

- Bus y Ethernet
- Anillo
- Árboles Binarios
- Estrella
- Mallas 1D
- Mariposa
- Pirámides
- Hipercubo
- Omega
- Fibras de interconexión

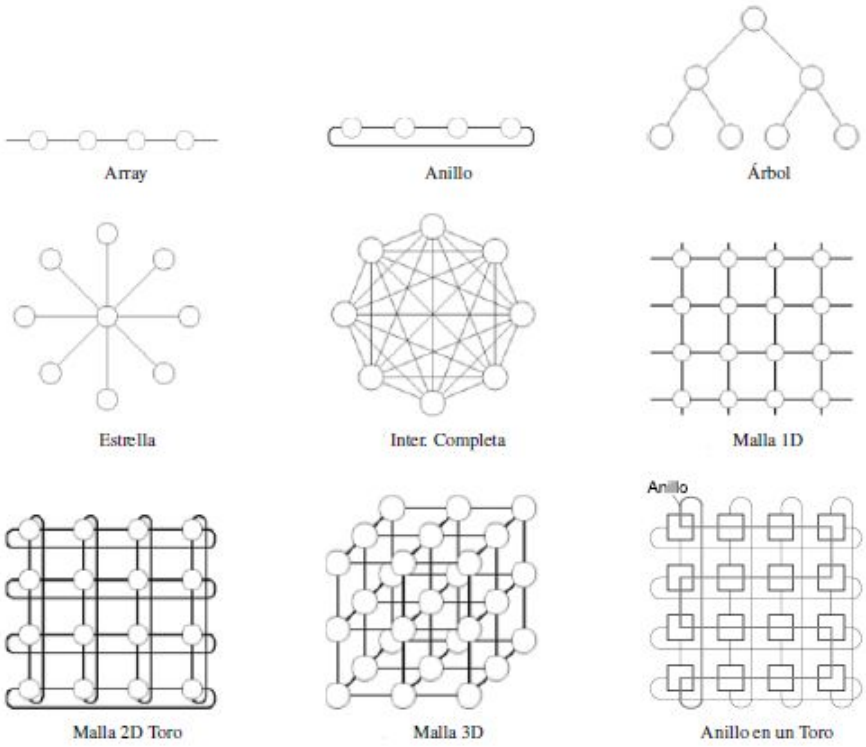

Figura 1.4: Diseño de redes de interconexión

La Tabla 1.1 muestra, para las topologías más empleadas, el máximo costo de un mensaje entre dos nodos cualquiera (Diámetro), la Conectividad que es una medida de la multiplicidad de caminos existentes entre dos nodos cualesquiera de la red; el número de enlaces de cada nodo (Grado) y el total de enlaces de la red (Costo).

Red	Diámetro	Conectividad	Grado	Costo
Conexión Total	1	$p-1$	$p-1$	$p(p-1)/2$
Estrella	2	2	$p-1$	$p-1$
Bus	$p-1$	1	3	$p-1$
Árboles Binarios	$2\log((p+1)/2)$	1	3	$p-1$
Mallas 1D	$2(\sqrt{p}-1)$	2	4	$2(p-\sqrt{p})$
Mallas 2D	$2(\sqrt{p})$	4	4	$2p$
Hipercubo	$\log p$	$\log p$	$\log p$	$(p\log p)/2$

Tabla 1.1: Características de algunas topologías de redes

1.4.1 Principales Arquitecturas Paralelas

Las principales arquitecturas paralelas que se emplean en la actualidad son:

Multiprocesadores o computadoras MIMD con memoria compartida.

Los procesadores comparten acceso a una memoria común, normalmente la comunicación es mediante un bus o una jerarquía de buses. Es una aproximación del modelo de computadora ideal conocido con el nombre de Parallel Random Access Machine (PRAM), donde cualquier procesador puede acceder a cualquier elemento de memoria en la misma cantidad de tiempo. En la práctica, la memoria suele estar jerarquizada; de esta manera se puede utilizar la frecuencia de acceso a los mismos datos almacenando copias de estos en la memoria caché asociada a cada procesador [22].

Multicomputadoras o computadoras MIMD con memoria distribuida.

En estos sistemas cada elemento de proceso tiene su propia memoria local donde almacena sus propios datos, no existiendo una memoria global común. Normalmente la comunicación es mediante una red de interconexión que une los distintos elementos de procesamiento entre sí y la comunicación entre los procesadores se realiza mediante paso de mensajes, donde se supone que la red de interconexión es ideal y por ello el costo de enviar un mensaje entre dos procesadores es independiente tanto de la localización del procesador como del resto del tráfico en la red, pero no de la longitud del mensaje [22].

Modelos Híbridos.

Los modelos híbridos tratan de obtener las ventajas de los modelos antes expuestos. Estos tipos de sistemas se caracterizan por organizar en grupos de elementos de cómputos

que comparten una misma memoria, la comunicación entre estos grupos se realiza mediante una red de interconexión. Siendo estos los modelos más usados en la actualidad, sobre todo por el desarrollo alcanzado en la fabricación de procesadores multinúcleos. Según el listado oficial de supercomputadoras más rápidas del mundo del 2010 existen 37 supercomputadoras que poseen esta arquitectura, esta lista esta encabezada por la Cray XT5-HE Opteron con un total de 224,162 núcleos [35], otros ejemplares bien conocidos son el NEC Earth-Simulator japonés con 5,120 CPUs organizadas en 640 grupos de 8 procesadores cada uno y una memoria total de 10,240 GB [16].

Clusters Beowulf

El desarrollo de sistemas operativos y compiladores del dominio público (Linux y GNU software), estándares para el pase de mensajes y conexiones universales a periféricos, han hecho posible tomar ventaja de los económicos recursos computacionales de producción masiva. La principal desventaja que presentan los proveedores de multiprocesadores es que deben satisfacer una amplia gama de usuarios, es decir, deben ser generales. Esto aumenta los costos de diseños y producción de equipos, así como los costos de desarrollo de software que va con ellos, sistema operativo, compiladores y aplicaciones. Cuando solo se necesita procesadores y un mecanismo de pase de mensajes no se debería pagar por todos estos añadidos que nunca usará. Estos usuarios son los que están impulsando el uso de clusters de PCs [22].

Un clúster Beowulf es un conglomerado de elementos de cómputo, gestionado por un sistema Unix, donde cada nodo es una computadora personal sin teclado, mouse, tarjeta de video o monitor. [9, 34] Aunque en un principio estos sistemas eran ignorados por su arquitectura de componentes poco acoplada ya han sido aceptados como un género de la comunidad HPC[2] [21].

Ventajas de usos de clusters de PCs para procesamiento paralelo.

- La reciente explosión en redes implica que la mayoría de los componentes necesarios para construir un cluster son vendidos en altos volúmenes y por lo tanto son económicos. Ahorros adicionales se pueden obtener debido a que solo se necesitará una tarjeta de vídeo, un monitor y un teclado por cluster. El mercado de los multiprocesadores es más reducido y más costoso.
- Remplazar un componente defectuoso en un cluster es relativamente trivial comparado con hacerlo en un multiprocesador, permitiendo una mayor disponibilidad en clusters cuidadosamente diseñados.

[2]Del Inglés High Performance Computing, Computación de Altas Prestaciones

Desventajas del uso de clusters de PCs para procesamiento paralelo

- Con raras excepciones, los equipos de redes generales producidos masivamente no están diseñados para procesamiento paralelo y típicamente su latencia es alta y los anchos de banda pequeños comparados con multiprocesadores. Dado que los clusters explotan tecnología que sea económica, los enlaces en el sistema no son veloces implicando que la comunicación entre componentes debe pasar por un proceso de protocolos de negociación lento, incrementando seriamente la latencia, por lo que suele recurrirse en el mejor de los casos a Fast Ethernet debido a los costos, restringiendo la escalabilidad del cluster.
- Existe poco soporte de software para manejar un cluster como un sistema integrado. La administración, por lo general, se debe hacer máquina por máquina y no es centralizada como en los multiprocesadores. Esto también lo hace más susceptibles a problemas de seguridad.
- Los procesadores para las computadoras personales no son tan eficientes como los procesadores de los multiprocesadores para manejar múltiples usuarios y/o procesos. Esto hace que el rendimiento de los clusters se degrade con relativamente pocos usuarios y/o procesos.
- Muchas aplicaciones importantes disponibles en multiprocesadores y optimizadas para ciertas arquitecturas, no lo están en clusters de computadoras personales.

Sin lugar a duda los clusters presentan una alternativa importante para varios problemas particulares, no solo por su economía, si no también por que pueden ser diseñados y ajustados para ciertas aplicaciones. Las aplicaciones que pueden sacar provecho de estos clusters de PCs deben estar caracterizadas por un grado de comunicación entre los componentes entre mediano y bajo o descompuestas en etapas independientes unas de las otras [22].

1.4.2 Diseño y Análisis de Algoritmos Paralelos

Diseñar algoritmos paralelos no es tarea fácil y es un proceso altamente creativo. Inicialmente se deben tener en cuenta los aspectos independientes de la máquina y los específicos a la máquina deben ser dejados para más tarde. El diseño involucra cuatro etapas las cuales se presentan como secuenciales pero en la práctica no lo son, según varios autores, en Grama 1994 [15] se plantean algunos pasos a tener en cuenta a la hora de diseñar algoritmos paralelos como:

- Identificar porciones de trabajo que pueden ser ejecutadas concurrentemente.
- Asignar los fragmentos concurrentes de trabajo en los procesadores que correrán en paralelo.
- Distribuir los datos de entrada, intermedios y de salida asociados con el programa.
- Sincronizar los procesadores durante las etapas de ejecución del programa paralelo.

Y otros autores como Foster [12] proponen la siguiente metodología a seguir (ver figura1.5)

1. Descomposición.
2. Comunicación.
3. Aglomeración.
4. Asignación.

Descomposición: El cómputo y los datos sobre los cuales se opera se descomponen en pequeñas tareas. Se ignoran aspectos como el número de procesadores de la máquina a usar y se concentra la atención en explotar oportunidades de paralelismo.

En la etapa de **Descomposición** se busca oportunidades de paralelismo y se trata de subdividir el problema lo más finamente posible, es decir; que la granularidad sea fina. Evoluciones futuras podrán llevar a aglomerar tareas y destacar ciertas posibilidades de paralelismo. Una buena partición divide tanto los cómputos como los datos. Hay dos formas de proceder con la descomposición [22].

1. **Descomposición del dominio**: el centro de atención son los datos. Se determina la partición apropiada de los datos y luego se trabaja en los cómputos asociados con los datos.
2. **Descomposición funcional**: es el enfoque alternativo al anterior. Primero se descomponen los cómputos y luego se ocupa de los datos.

Comunicación: Se determina la comunicación requerida para coordinar las tareas. Se definen estructuras y algoritmos de comunicación.

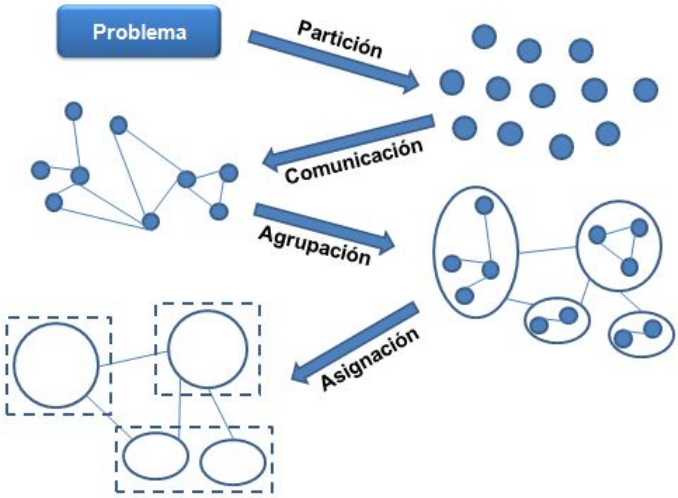

Figura 1.5: Etapas en el diseño de algoritmos paralelos

Aglomeración: El resultado de las dos etapas anteriores es evaluado en términos de eficiencia y costos de implementación. De ser necesario, se agrupan tareas pequeñas en tareas más grandes.

Asignación: Cada tarea es asignada a un procesador tratando de maximizar la utilización de los procesadores y de reducir el costo de comunicación. La asignación puede ser estática o en tiempo de ejecución mediante algoritmos de balanceo de carga.

1.4.3 Principales parámetros para medir el desempeño de algoritmos paralelos.

Aceleración y Eficiencia

Dos de las medidas más importantes para apreciar la calidad de la implementación de un algoritmo paralelo en multi-computadores y multiprocesadores son aceleración[3] y eficiencia. La aceleración de un algoritmo paralelo ejecutado usando p procesadores es la razón entre el tiempo que tarda el mejor algoritmo secuencial en ser ejecutado en el mismo computador usando p procesadores. La eficiencia de un algoritmo paralelo ejecutado en p procesadores es la aceleración dividida por p. La ley Amdahl permite

[3]El término más empleado es Speed-Up de su significado en inglés

expresar la aceleración máxima obtenible como una función de la fracción de código del algoritmo que se pueda paralelizar [15, 22]. Como consecuencia de la misma se tiene que al aumentar el número de procesadores para un mismo tamaño del problema provoca una disminución de la eficiecia del algoritmo.

Ley de Amdahl

$$E = \frac{1}{1 + T_O/T_s} \tag{1.3}$$

Donde T_O es el tiempo de solapamiento y T_S el tiempo del mejor algoritmo secuencial. El tiempo de solapamiento aumenta conforme lo hace p, note que todo algoritmo paralelo tiene un componente secuencial C_S, cuya ejecución toma un tiempo t_{sec}, mientras un procesador ejecuta C_S los restantes $p-1$ procesadores estarán ociosos durante el tiempo t_{sec} por lo que T_O tendrá, al menos, un componente igual a $(p-1) \times t_{sec}$ lineal respecto a p.

Eficiencia

Si se normaliza el Speed-up dividiéndolo por el número de procesadores se obtiene la eficiencia, que expresa el grado de utilización de un sistema multiprocesador [15].

$$E_p = \frac{S_p}{p} \tag{1.4}$$

Donde S_p es el Speed-up del algoritmo y p el número de procesadores.

Costo de Comunicación

Considerando que el código sea 100 % paralelizable, por tanto el Speed-up es, $S_\mu = p$.

Despreciando el tiempo de comunicación entre procesadores. Considerando el tiempo de comunicación o latencia. T_C, el Speed-up decrece aproximadamente a,

$$S_p = \frac{T}{t/p + T_c} < p \tag{1.5}$$

Donde T es el tiempo que tarda el mejor algoritmo secuencial en ser ejecutado por

un computador. Para que el Speed-up sea afectado por la latencia de comunicación se necesita que,

$$\frac{T}{p} \gg T_c \Rightarrow p \ll \frac{T}{T_c} \tag{1.6}$$

Esto significa que a medida que se divide el problema en partes más pequeñas para poder ser ejecutado en más procesadores, llega un momento en que el costo de comunicación T_c se hace muy significativo lo que provoca una desaceleración en el proceso de cómputo.

Escalabilidad

Un sistema es escalable si es posible mantener la eficiencia constante incrementando simultáneamente el tamaño del problema w definido como la complejidad del algoritmo que lo soluciona y el número de procesadores p usados para su ejecución. Por lo que la escalabilidad de un sistema es una medida de su capacidad para incrementar el Speed-up en proporción al número de procesadores, reflejando la capacidad del mismo de utilizar de forma efectiva nuevos recursos que les sean incorporandos. [15].

1.5 Librería de Paso de Mensaje

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. Donde el modelo de programación a emplear para la implementación de un algoritmo paralelo en gran medida queda determinado por el hardware a emplear para su ejecución y de las herramientas que se empleen. Para el caso que la solución se implemente con el objetivo de ser ejecutado sobre un cluster de computadoras con memoria distribuida existen varios estándares como MPI (Message Passing Interface), PVM (Parallel Virtual Machine), CILK, C*, HPF y ZPL, donde los más usados para el desarrollo del modelo de programación mediante paso de mensajes son MPI y PVM.

Máquina Virtual Paralela(PVM)

La Máquina Virtual Paralela (PVM) es una biblioteca para el cómputo paralelo en un sistema distribuido de computadoras. Está diseñado para permitir que una red de computadoras heterogénea comparta sus recursos de cómputo como el procesador y la memoria RAM con el fin de aprovechar esto para disminuir el tiempo de ejecución de un programa al distribuir la carga de trabajo en varias computadoras.

La biblioteca PVM fue desarrollada por la Universidad de Tennessee, en el Laboratorio Nacional Oak Ridge y la Universidad Emory. La primera versión fue escrita en ORNL en 1989 y el objetivo general de este proyecto es investigar en temas relacionados con la computación de altos rendimientos y desarrollar soluciones para sistemas informáticos concurrentes heterogéneos. Esta biblioteca es un conjunto integrado de herramientas de software y librerías que emulan un heterogéneo marco de computación simultánea, en ordenadores interconectados de variada arquitectura, de forma general y flexible [2].

Principales Características de PVM.

- Heterogeneidad
- Escalabilidad
- Múltiple representación de datos
- Tolerancia ante fallos.

Paralelismo con paso de mensajes: MPI

La descripción de más alto nivel de un cluster se corresponde con computadoras autónomas conectadas mediante una red. La forma natural de programación a este nivel consiste en procesos independientes que colaboran intercambiando mensajes.

Desde la aparición de los primeros multicomputadores se han desarrollado múltiples métodos de programación con pasos de mensajes como Occam, PVM y MPI. Entre todos ellos MPI se ha convertido en un estándar ampliamente utilizado y soportado en prácticamente todas las plataformas disponibles.

MPI consiste en un entorno de ejecución y una librería de programación. El entorno de ejecución ofrece los servicios requeridos para acceder al soporte hardware de comunicación y la librería proporciona la interfaz de programación para iniciar los procesos y realizar las comunicaciones. El entorno de ejecución y su interfaz puede diferir ligeramente entre las diferentes implementaciones, sin embargo la interfaz de programación está totalmente estandarizada de forma que los programas basados en MPI son totalmente portables.

Las características principales de la interfaz de programación que proporciona MPI son comunicación punto a punto síncrona y asíncrona, comunicaciones colectivas y topologías virtuales. La versión 2 del estándar añade capacidad multihilo, creación dinámica de procesos, entrada-salida paralela y comunicación unidireccional. Este último aspecto es particularmente novedoso ya que aproxima MPI al modelo de memoria compartida. Se trata de que un proceso pueda acceder a variables ubicadas en otro proceso sin que este último intervenga [19].

Características generales de MPI

Las implementaciones de MPI están disponibles principalmente para el lenguaje C. Aunque también existen extensiones para C++ y las implementaciones para Fortran son bastante comunes y en la actualidad también se pueden encontrar implementaciones para Java.

La estructura básica de un programa paralelo que utilice MPI tendrá que realizar las siguientes operaciones:

1. Inicializar el entorno de MPI.
2. Comunicación entre procesos.
3. Finalizar el entorno de MPI.

La versión 1.0 del estándar MPI define 125 funciones. Sin embargo, las funciones básicas, para realizar las principales operaciones anteriores, son las que se pueden observar en la tabla1.5.

Tipo de función	Nombre	Acción
Inicialización del entorno	MPI_Init()	Inicializa el entorno de MPI
	MPI_Comm_size()	Retorna el número de procesos
	MPI_Comm_rank()	Retorna el identificador de cada proceso
Comunicación	MPI_Send ()	Envía un mensaje
	MPI_Recv()	Recibe un mensaje
Finalización del entorno	MPI_Finalize()	Termina el entorno de ejecución paralela.

Tabla 1.2: Principales Funciones de MPI

El envío de mensajes entre procesos MPI se realiza por medio de comunicadores. Un comunicador es una colección de procesos que comparte el envío y recepción de mensajes. Por defecto MPI crea un comunicador general que incluye todos los procesos disponibles

en el entorno paralelo, llamado *MPI_COMM_WORLD*. MPI proporciona funciones que permiten al usuario, a partir de este comunicador, crear comunicadores con distintos procesos. Los comunicadores se usan para restringir los mensajes a los procesos dentro del comunicador. Los mensajes enviados entre procesos de un comunicador pueden ser punto a punto o colectivos. Las funciones que MPI provee para comunicaciones colectivas son:[12]

- MPI_Bcast()
- MPI_Scatter()
- MPI_Gather()
- MPI_Allgather()
- MPI_Alltoall()
- MPI_Reduce()

Son pocas las aplicaciones actuales que aprovechen el poder de cómputo adicional que proporciona el hardware, por lo que los programadores deberán prepararse para que puedan crear nuevos tipo de aplicaciones, usando la programación paralela y elevando los niveles de aprovechamiento de los recursos disponibles [19].

2

Descripción y Análisis de la Solución Propuesta

En este capítulo se presenta la solución propuesta al problema que origina esta investigación. Inicialmente se realiza un análisis desde el punto de vista secuencial para luego dar paso a un algoritmo paralelo que se propone como solución al problema. Del cual se muestra un análisis teórico de los principales parámetros que miden su desempeño. Además de abordar el diseño del sistema y las librerías necesarias para su implementación.

2.1 Algoritmo Secuencial

Como se ha venido analizando desde el capítulo anterior a partir de realizar el Muestreo al yacimiento y con este obtener la información necesaria para realizar la etapa de Determinación de Clase donde se obtienen las principales características de las muestras, dando como resultado una tabla que contiene por cada muestra los campos (pozo, bloque y profundidad) que describen geoespacialmente el punto de muestreo y un campo con la clase patrón observada, una vez en este estado, se aplica el modelo 1.2 al muestreo realizado. El Algoritmo Secuencial 2.1.1 es mediante el cual se obtiene el modelo.

Algorithm 2.1.1: Algoritmo Secuencial

Data: Registro C de n muestras
Result: Hipercubo de probabilidades de transición M

```
1 M ← ∅;
2 for i = 1 to n − 1 do
3     for j = i + 1 to n do
4         variables_modelo = comparar(c_i, c_j);
5         if Influencia(variables_modelo) then
6             ajustarModelo(M, variables_modelo) ;
7 return M ;
```

Un concepto importante a tener en cuenta presente en el algoritmo es la comparación entre dos registros c_i y c_j, la cual es para reflejar en el modelo la influencia que tiene la clase patrón del registro c_i sobre la clase patrón del registro c_j y viceversa. Además es valido señalar que esta operación se realiza mediante el procedimiento $comparar(c_i, c_j)$ el cual no es conmutativo, pues la característica observada en c_i no influye sobre la observada en c_j de la misma forma en que lo hace c_j sobre c_i [23] . Esto trae como consecuencia que por cada comparación se determinan los valores de cada uno de los parámetros que indican los dos elementos de M a modificar (ver algoritmo 2.1.2). Las principales funciones que intervienen en este proceso son descritas a continuación.

La función $Influencia(variables\_modelo)$ determina a partir de la distancia entre las muestras a comparar, previamente calculada en la comparación de c_i, c_j, si la clase característica presente en c_i puede condicionar la presencia de c_j y viceversa, lo cual se considera cierto si ambas están separadas a una distancia no mayor que una cota previamente fijada.

La función $ajustarModelo(\mathrm{M}, variables\_modelo)$ tiene el objetivo de modificar en M las probabilidades de transición del estado c_i al c_j y viceversa a partir de los valores

$\{H, P, \alpha_H, \alpha_{H1}, \alpha_V, \alpha_{V1}, c_i.clase, c_j.clase\}$. Esta función influye de forma significativa en el desempeño del algoritmo, pues a pesar que su ejecución toma un tiempo constante respecto al tamaño de la entrada, el aumento de las dimensiones del hipercubo hacen que su complejidad aumente también.

Algorithm 2.1.2: Función de Comparar (c_i, c_j)

```
1 comparar(c_i, c_j);
2 H = profundidad(c_i, c_j);
3 P = paso(c_i, c_j);
4 α_H = direccionHorizontal(c_i, c_j);
5 α_H1 = direccionOpuesta(α_H);
6 α_V = direccionVertical(c_i, c_j);
7 α_V1 = direccionOpuesta(α_V);
8 return {H, P, α_H, α_H1, α_V, α_V1} ;
```

Cada una de las 6 *dimensiones* de M son: característica observada en la muestra origen $c_i.clase$, característica observada en la muestra destino $c_j.clase$, profundidad de la muestra H, distancia entre las muestras tomadas P, dirección horizontal de la muestra destino respecto a la muestra origen α_H y dirección vertical de la muestra destino respecto a la muestra origen α_V. Cuando se realiza una comparación las características observadas están presentes en el registro, por lo que $comparar(c_i, c_j)$ se limita al cálculo de los otros 4 índices en ambos órdenes.

La profundidad H de una comparación se determina como el máximo de las profundidades de las muestras a comparar. El paso P, que expresa la distancia entre las muestras a comparar, se define como la cantidad de pozos que separa ambas muestras dentro de la red de exploración y se determinarse como la Distancia de Chebyshev: $P = \text{máx}(|X_1 - X_2|, |Y_1 - Y_2|)$. Cuando se comparan muestras pertenecientes a un mismo pozo entonces el paso se determina como la diferencia, en metros, de las profundidades. El índice α_H, que representa la dirección horizontal de la muestra destino c_j respecto a la muestra origen c_i, expresa la dirección en función de una de las 16 direcciones de la rosa náutica donde el ángulo 0 fija el Oeste y π el Este. La dirección vertical α_V, se calcula de forma análoga, pero reduciéndola a solo 6 sectores.(ver Anexo A)

Este algoritmo fue implementado y para problemas de pequeño tamaño el algoritmo realiza el modelado en un tiempo adecuado, sin embargo, para problemas de medio y gran tamaño, aun sigue siendo insuficiente el tiempo de respuesta. Por lo que aun no es posible su incorporación al proceso ORENI[1].

[1]Optimización de Redes de Exploración de Níquel

2.2 Algoritmo Paralelo

El Algoritmo 2.2.1 propuesto por [36] es el utilizado para la implementación del sistema, el cual está basado en una descomposición funcional mediante asignación de bloques extremos de $n/2p$ registros consecutivos.

Esta descomposición funcional fue la más acorde al problema, en la misma la tabla de entrada C de n registros es dividida en $2p$ bloques de igual tamaño, cada uno formado por $n/2p$ registros consecutivos donde p es el total de procesadores y de esta forma todos los procesadores tienen la misma carga, al menos durante la ejecución de esta primera fase del algoritmo dedicada a la comparación de los registros.

Una vez realizada las comparaciones, en cada procesador p_k se tiene una matriz M_k^0, por lo que es necesario realizar la combinación de todas para obtener la Matriz de Probabilidades de Transición final $M = \sum_{i=0}^{p-1} M_i^0$. Este proceso puede realizarse mediante una suma en forma de árbol, donde los nodos hojas son las matrices correspondientes a cada uno de los p procesadores y la matriz resultado es la suma de las matrices obtenidas en los procesadores p_k y p_r con $r = k + p - 2^{j-1}$ MOD p en varias iteraciones. De esta forma al final del proceso, en el $\log p$ paso, en p_0 quedará la matriz resultante la cual es el objetivo del algoritmo.

Algorithm 2.2.1: Algoritmo Paralelo

Data: Registro C de n muestras
Result: Hipercubo de probabilidades de transición M

$M \leftarrow \emptyset$;
En Paralelo:
for $pr = 1, 2, 3 \ldots, p-1$ **do**
  En pr:
  //Fase I Cada procesador pr calcula su matriz M^0_{pr}
  /* Registros del bloque B_{pr} que conforman a t_{pr}*/
  for $i = pr\ n/2p$ **to** $(pr+1)n/2p$ **do**
    for $j = i+1$ **to** n **do**
      $variables\_modelo \leftarrow comparar(c_i, c_j)$;
      if $Influencia(variables\_modelo)$ **then**
        $ajustarModelo(\mathrm{M}, variables\_modelo)$;
  /* Registros del bloque $B_{2p-pr-1}$ que conforman a t_{pr}*/
  for $i = n(1-(pr+1)/2p)$ **to** $n(1-pr/2p)-1$ **do**
    for $j = i+1$ **to** n **do**
      $variables\_modelo \leftarrow comparar(c_i, c_j)$;
      if $Influencia(variables\_modelo)$ **then**
        $ajustarModelo(\mathrm{M}, variables\_modelo)$;
  //Fase II Cálculo de $M = \sum_{i=0}^{p-1} M_i^{\log p}$
  /* En cada paso $\log p$ i-ésimo se realizan $p/2^i$ sumas acumulativas */
  for $i = 1$ **to** $\log n$ **do**
    if $pr\ MOD\ 2^i = 0$ **then**
      Recibir$Matriz$;
      $M^i_{pr} = M^{i-1}_{pr} + Matriz$;
    else
      Enviar$M^i_{pr} \Rightarrow p_k$ $//k = (pr + 2^{i-1})\ MOD\ p$;
      return 0;
  if $pr = 0$ **then**
    return $M^{\log p}_{pr}$;

2.2.1 Análisis analítico del algoritmo

Para el Algoritmo 2.2.1 en [36] se determinaron las principales métricas que permiten evaluar un algoritmo paralelo. El primer parámetro analizado fue el tiempo paralelo del algoritmo T_P el cual para sistemas paralelos con memoria distribuida se determina según [15], como $T_P = T_A + T_C - T_{SOL}$ donde T_A es el tiempo aritmético, T_C el tiempo de comunicación y T_{SOL} es el tiempo de solapamiento. El mismo para el algoritmo 2.2.1 está regido por la ecuación 2.1.

$$T_P = \frac{cn^2}{2p} + m \log p + (2(p-1))(mt + \beta) \text{ Flops} \tag{2.1}$$

Donde t es el tiempo de envío de una palabra, β el tiempo de latencia y m es la cantidad de elementos de la matriz.

Según la definición de *Speed-up* (S_p) y la Eficiencia E_p propuesta en la sección 1.4.3 por [15] para el algoritmo 2.2.1 se determinó en [36] los resultados correspondientes:

$$\lim_{n\to\infty} S_p = p \tag{2.2}$$

$$\lim_{n\to\infty} E_p = 1 \tag{2.3}$$

Otro parámetro muy importante de analizar en el algoritmo es el Desequilibrio de Carga el cual según [4, 15] expresa cuán equilibrado está el trabajo entre los p procesadores del sistema y para el algoritmo 2.2.1 en [36] se determinó que está dado por la expresión:

$$D_q = m \log p \ Flops \tag{2.4}$$

El número óptimo de procesadores según [15] puede determinarse haciendo que derive un tiempo paralelo mínimo, partiendo del T_P en [36] se determinó que para el algoritmo 2.2.1 está dado por la expresión 2.5, aunque en la práctica como es evidente se usarán los procesadores con los que se cuente.

$$p = \frac{\sqrt{m^2 + 4cn^2(mt + \beta)} - m}{4(mt + \beta)} \tag{2.5}$$

2.3 Implementación

En la descripción del algoritmo se propone una estrategia general que en función del tamaño del problema persigue obtener ganancia de tiempo, sin embargo como puede verificarse en el tiempo paralelo 2.1 hay otros parámetros que pueden influir en su rendimiento, tal es el caso del costo de realizar una comparación c y la cantidad de elementos m de cada uno de los hipercubos obtenidos en la *Fase I* del algoritmo, la cual determina directamente el tiempo requerido para sumar dos de estas, algunas consideraciones de estos parámetros serán analizadas con el objetivo de optimizar la implementación.

El modelo es representado mediante un hipercubo de probabilidades condicionales, que se obtiene de la suma de cada uno de los hipercubos calculados en cada procesador. Si estos son representados en memoria como bloques lineales de elementos por cada dimensión daría lugar a una representación clásica de una matriz (*Variante Clásica*) donde la cantidad de elementos de cada una de ella es la misma e igual a la multiplicación de cada una de las dimensiones que la conforman.

$$m = |\alpha_H| \times |\alpha_V| \times |H| \times |P| \times |k| \times |k| \tag{2.6}$$

Por lo tanto el tiempo de sumar dos hipercubos cualquiera está acotado por un tiempo $T_m = m$. Teniendo en cuenta que este hipercubo almacena de cierta forma la frecuencia de aparición de una característica del suelo dada otra condicionada por 4 variables adicionales, cabe cuestionarse cuán frecuente es no encontrar dependencia alguna para valores específicos de estas, lo que puede dar lugar a elevados valores de dispersión, o sea una gran cantidad de elementos iguales a cero. Teniendo en cuenta además las dimensiones de este hipercubo puede verse que hay un requerimiento importante de memoria. Teniendo en cuenta estos elementos se puede analizar la posibilidad de solo almacenar los elementos que son distintos de ceros, sin perder la referencia de a donde pertenecerían dentro del hipercubo original.

Una implementación del algoritmo siguiendo esta característica será más eficiente en cuanto a tiempo y espacio sólo cuando la suma de dos matrices representadas de esta forma tome un tiempo menor o igual que T_m, manteniendo el acceso a cada elemento constante, y el total de espacio necesario para su almacenamiento sea menor que el requerido para almacenar los m elementos del hipercubo original.

Supóngase que los hipercubos son representados como colección de pares de la forma (*Elemento*, *Coordenadas*), así el componente Coordenadas debe ser un valor capaz de almacenar los 6 índices que expresan el lugar que ocupa el *Elemento* dentro del hipercubo

original, la cual se obtiene mediante un proceso matemático que de lugar a un valor único que represente esta coordenada y mediante un proceso inverso se pueden obtener nuevamente los índices que le dieron origen.

Suponiendo que el total de elementos distintos de cero es m_c, esta representación es más eficiente (respecto a la Variante Clásica) en cuanto a espacio ssi $m_c < m/2$. Sin embargo el tiempo para acceder a un determinado elemento no es constante, pues se requiere su búsqueda a lo largo de la colección, que si es representada como una lista entonces es $T_{m_c} = O(m_c)$ pero si se representa como un árbol AVL[2] (*Variante AVL*) entonces el tiempo es $T_{m_c} = O(\log m_c)$.

En la figura 2.1 se muestra un hipercubo de 3 dimensiones mediante la Variante AVL.

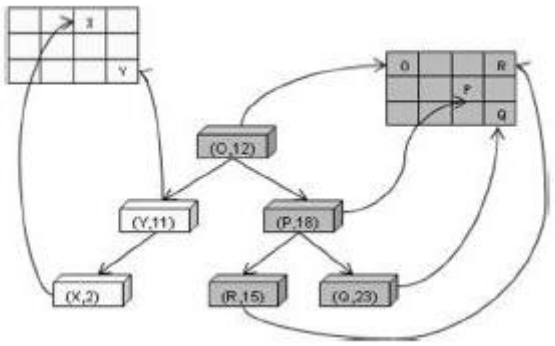

Figura 2.1: Variante AVL

Como ya se mencionaba, en la *Fase II* del algoritmo se realizan la combinación de los resultados de la *Fase I*, que siguiendo la Variante Clásica toma un tiempo T_m, pero si se escoge la Variante AVL es $T_{m_c} = O(m_c \log m_c)$, lo que establece que esta *Fase II* del algoritmo será más rápida solo cuando $m_c \log m_c < m$, de aquí que se puede decir que la Variante AVL es más óptima, en cuanto a tiempo respecto a Variante Clásica, siempre que se cumpla que m_c sea a lo sumo un $x\,\%$ de m y en cuanto a espacio siempre que $m_c < m/4$ pues se debe almacenar por cada nodo del árbol, el valor de la probabilidad, su posición dentro del hipercubo, además de los dos punteros a los nodos hijos.

Para tener un menor gasto de memoria se puede utilizar una implementación de la matriz en forma de un arreglo ordenado denominada (Versión Lineal o Vector Ordenado), siguiendo el mismo principio de la versión AVL para el cálculo de las coordenadas de un elementos pero en este caso los elementos serán almacenados de forma lineal mediante un

[2] Árbol Binario de Búsqueda Balanceado ideado por los matemáticos rusos Adelson-Velskii y Landis

arreglo de nodos (*Elemento*, *Coordenadas*). Si estos valores mayores que cero representan el $x\,\%$ del total de elementos de la matriz entonces el gasto de memoria sería $(4 * \prod_{i=1}^{6} dim_i) * \frac{x}{100} * 2$ bytes, donde dim_i es la cardinalidad de la dimensión i, y se multiplica por 4 porque el tamaño de un entero es 4 bytes.

En la figura 2.2 se muestra un hipercubo de 3 dimensiones mediante la Variante Vector Ordenado.

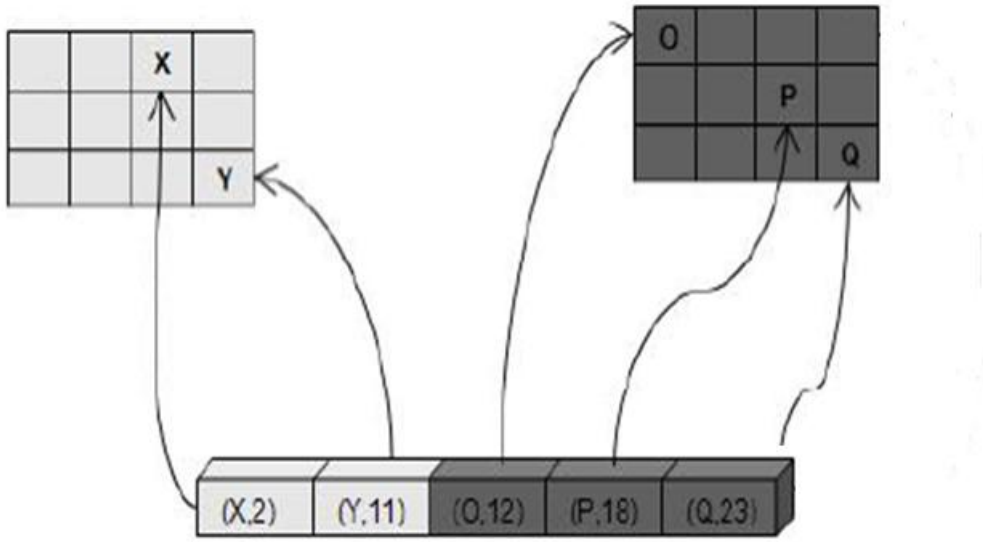

Figura 2.2: Variante Vector Ordenado

Del análisis anterior se deduce que sólo es aplicable para $x \leq 50$. Pero esta variante trae consigo que aumente el tiempo de ejecución ya que las modificaciones tendrían complejidad $O(\log m_c)$ y las inserciones $O(m_c)$ para el peor de los casos ya que esta operación conlleva tener que realizar copias de todos los elementos una vez que se ha agotado la memoria reservada y existe la necesidad de agregar un nuevo elemento. Para el caso promedio en que aun hay capacidad de memoria disponible la operación de inserción tendrá una complejidad temporal de $O(\log m_c)$.

A continuación se analiza el consumo de memoria y complejidad temporal de la Variante Lineal con respecto a las dos variantes anteriores (Clásica y AVL), esta variante tiene un gasto de memoria menor que la Variante Clásica, la cual lo tiene determinado por $4 * \prod_{i=1}^{6} dim_i$, y con respecto a la variante AVL el consumo de memoria es equivalente a $1/2$ de esta; en cuanto a la complejidad temporal de la Variante Lineal es mayor pero muy cercana a la Variante AVL debido a que ambas tienen la misma complejidad en la operación de modificación siendo sólo diferente la complejidad para la operación de

inserción siendo $O(m_c)$ y $O(\log m_c)$ respectivamente para la Variante Lineal y AVL en el caso de la Variante Clásica la cual tiene una complejidad de $O(1)$ para sus operaciones esta Variante Lineal representa una solución de mayor complejidad.

Para la Variante AVL y Lineal la complejidad temporal como se pudo observar estará determinada entonces por la ocurrencia de la operación de inserción, en la medida que se hagan más inserciones que modificación al modelo sera más costosa temporalmente el empleo de la Variante Lineal con respecto a la Variante AVL.

Como conclusión de esta sección se puede plantear que la Variante AVL, la cual emplea 4 enteros por cada elemento como se explicó anteriormente es la variante más equilibrada en cuanto al consumo de memoria y tiempo. La misma debe ser empleada 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 $4 * \prod_{i=1}^{6} dim_i * \frac{x}{100} * 4$, 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 $O(\log m_c)$, entonces es más eficiente temporalmente en comparación con la implementación que utiliza el Vector Ordenado o Variante Lineal aunque no lo sea con respecto a la Variante Clásica.

2.3.1 Herramientas utilizadas

Para la implementación del sistema se utilizó el IDE de desarrollo Code:Blocks en su versión 8.2, el mismo es muy empleado para la programación en C y C++, por sus características de ser ligero, multiplataforma, reconoce varios compiladores para dichos lenguajes de programación.

2.3.2 Implementación Paralela empleando MPI

Para la implementación del algoritmo paralelo propuesto en esta investigación se usó MPI desde el lenguaje C. Algunas características que determinaron su elección fueron el soporte para el modelo de programación SPMD, elevados desempeños para la comunicación basados en su conjunto de funciones potentes y flexibles, API extenso que permite las más disímiles operaciones, sin embargo con un conjunto pequeño de funciones pueden programarse un amplio rango de problemas.

En el algoritmo 2.2.1 anterior no se tuvo en cuenta el tipo de dato que tiene la matriz pues el presente trabajo considera tres variantes a la hora de representar el modelo 1.2

descrito por la matriz con alguna estructura de datos.

La implementación del sistema está compuesta por los siguientes ficheros fuentes:

- Algoritmo.h
- arbolfuente.h
- Negocio.h
- TiposDatos.h
- UtilesAlgoritmo.h
- UtilesMPI.h
- Algoritmo.c
- arbolfuente.c
- IO.c
- Negocio.c
- UtilesAlgoritmo.c
- UtilesMPI.c
- main.c

En *Algoritmo(.h .c)* está presente la implementación de la obtención del modelo 1.2 para cada una de las posibles variantes de almacenamiento descritas. Un ejemplo de la declaraciones para la variante AVL es la siguiente:

```
AVLTree* ModeloUsandoArbol(Registro* records, int my_id, int nproc,
 int n ,double* TiempoProcesamiento, double* TiempoTransferencia);
```

Los parámetros necesarios para la obtención del modelo son:

```
Registro* records //Lista de registros iniciales.
int my_id //Identificador del proceso.
int nproc //Cantidad de procesos
int n,    //Cantidad de Registros
double* TiempoProcesamiento //Tiempo de Procesamiento (segundos)
double* TiempoTransferencia //Tiempo necesario para las sumas
                            //de los Hipercubos (segundos)
```

En *Negocio(.h .c)* está presente la implementación de las principales funciones necesarias para realizar las comparaciones entre los registros y poder obtener el modelado además de la cardinalidad de las dimensiones del modelo.

```
// Cardinalidad de las dimensiones del modelo.

#define DIMCARACTERISTICAS 16
#define DIMPASO 9
```

```
#define DIMPROFUNDIDAD 52
#define DIMANGHORIZONTAL 17
#define DIMANGVERTICAL 6
```

```
/* Principales funciones para el cálculo del modelo */

int paso(Pozo p0, Pozo p1);
int dirHoriz(Pozo orig, Pozo dest);
int dirVert(Pozo p0, Pozo p1, double hOrig, double hDest);
double swAngulo(double dx, double dy);
```

En la función *comparar*(c_i, c_j)2.1.2 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 *int dirHoriz(Pozo orig, Pozo dest)* no es más que el cálculo del 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. Ver figura2.3 y Anexo A.

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 $\alpha = \frac{2\pi}{n}$ donde n depende de la exactitud a la que se quiera calcular el modelo y siempre los sectores se ubican a partir del ángulo cero y en sentido contrario a las manecillas del reloj.

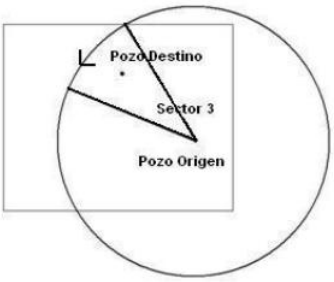

Figura 2.3: Sector del ángulo horizontal

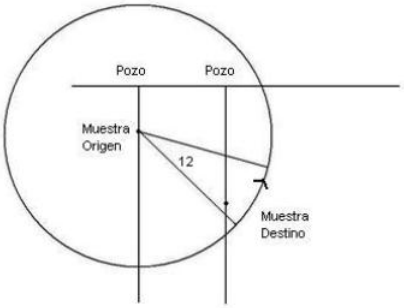

Figura 2.4: Sector del ángulo vertical

El sector del ángulo vertical que se obtiene en la función *int dirVert(Pozo p0, Pozo p1,*

double hOrig, double hDest) 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 donde n depende de la exactitud a la que se quiera calcular el modelo.

En *UtilesMPI(.h .c)* está presente la implementación de las funciones que hacen uso de la librería MPI para la distribución inicial de los registros a ser comparados por cada uno de los nodo de procesamiento y además las funciones que intervienen en el envío de los hipercubos entre los nodos de procesamiento para realizar las sumas acumulativas una vez concluidas las comparaciones.

```
void DistribucionDatos_Nodos(Registro* records, int CantRegistros,
                             int my_id);
void EnviarArbol(AVLTree *t,MPI_Datatype datatype, int dest,
                 int tag, MPI_Comm comm);
void EnviarLista(Lista *nodo,MPI_Datatype datatype, int dest,
                  int tag, MPI_Comm comm);
void EnviarMatriz(int ******matriz,MPI_Datatype datatype, int dest,
                     int tag, MPI_Comm comm);
```

A continuación se muestra un fragmento de la implementación de *EnviarArbol(...)* y *DistribucionDatos_Nodos(...)* en las cuales se puede evidenciar el empleo de funciones de MPI como *MPI_Bcast* y *MPI_Send* en el Anexo B puede encontrase la implementación integra.

Función para la distribución inicial de los registros.

```
void DistribucionDatos_Nodos(Registro* records, int CantRegistros,
                             int my_id){
{...}
for(j =0; j<CantRegistros ;j++)
 {
     if (my_id == 0){
      val_reg[0] = records[j].posBloque;
      {....}
     }
   MPI_Bcast(val_reg,4,MPI_INT,0,MPI_COMM_WORLD);
    {...}
 }
}
```

Función para realizar las sumas acumulativas de los Hipercubos.

```
void EnviarArbol(AVLTree *t,MPI_Datatype datatype,  int dest, int tag,
                 MPI_Comm comm)
{
  {...}
  if(t->key)
  {
   nodo[0] = t->key;
   nodo[1] = t->dato;
   MPI_Send(nodo,2,datatype,dest,tag,comm);
  }
  {...}
}
```

A continuación se muestra una síntesis de las dos funciones más importante presentes en el sistema, la función encargada de realizar la comparación entre los registros y la obtención del modelo, en este caso se verá la implementación para la variante AVL.

Función para comparar dos registros.

```
void CompararRegistros(Indices* IndicesOD1, Indices* IndicesDO2
,Registro recordI, Registro recordF)
{
{...}
POrigen.posNS = nortSur_EstOst(POrigen.posGlobal);
POrigen.posEO = nortSur_EstOst2(POrigen.posGlobal);
indicesOD.CaractOrigen  = recordI.posCaracteristica;
indicesDO.CaractDestino = recordI.posCaracteristica;
{...}
PDestino.posNS = nortSur_EstOst(PDestino.posGlobal);
PDestino.posEO = nortSur_EstOst2(PDestino.posGlobal);
indicesOD.CaractDestino = recordF.posCaracteristica;
indicesDO.CaractOrigen = recordF.posCaracteristica;
{...}
int dh = dirHoriz(POrigen, PDestino);
indicesOD.DireccHorz = dh;
indicesDO.DireccHorz = (dh > 0)?((dh+8)%16):dh;

indicesOD.DireccVert = (dirVert(POrigen, PDestino, recordI.posProfundidad,
    recordF.posProfundidad) - 1);
indicesDO.DireccVert = (dirVert(PDestino, POrigen, recordF.posProfundidad,
    recordI.posProfundidad) - 1);
{...}
indicesOD.Paso = indicesDO.Paso = paso(POrigen, PDestino);
{...}
```

```
*IndicesOD1 = indicesOD;
*IndicesDO2 = indicesDO;
}
```

Función para obtener el modelo en una representación AVL

```
AVLTree * ModeloUsandoArbol(Registro *records, int my_id, int nproc,
    int n, double *TiempoProcesamiento, double* TiempoTransferencia)
{
{...}
//Comienzo la comparación de los registros en Paralelo
/* Cálculo el inicio y fin de cada Bloque de
acuerdo al proceso $B_{r}$ y $B_{2p-r-1}$ */

 int size = n/(2*nproc);
 int startRecord = my_id*(n/(2*nproc));
 int indPartSegBloq = n - startRecord - size;
 int indPartPriBloq = startRecord;
 int limiteBloq1 = startRecord + size;
 int limiteBloq2 = indPartSegBloq + size;
 {...}
    for( i= indPartPriBloq ; i < limiteBloq1 ; i++)
    {
     for (j = i + 1; j < n; j++)
      {
        CompararRegistros(&indicesOD, &indicesDO, records[i], records[j]);

          if (coordOK(indicesOD) == 1 && coordOK(indicesDO) == 1)
          {
           posOD = posicion(indicesOD);
           posDO = posicion(indicesDO);
           insertar(&arbol,posOD, 1);
           insertar(&arbol,posDO, 1);
           insertar(&arbol,posOD - indicesOD.CaractOrigen + 16, 1);
           insertar(&arbol,posDO - indicesDO.CaractOrigen + 16, 1);
          }
      }
     }
     //Comparación de la registros de $B_{2p-r-1}$
     {...}
     // Fin de la comparación entre los registros

     /* Envío y suma de los Hipercubos almacenados
     en cada árbol AVL presente en cada procesador. */

    for (i =1; i <= Logaritmo2(nproc); i++)
    {
```

```
    if( (my_id%((int)pow(2,i)) ) == 0)
     {
      k = (int)(my_id + (nproc - pow(2,i-1)))\% nproc;
       {...}
       MPI_Recv(nodo,2,MPI_INT,k,99,MPI_COMM_WORLD,&status);
       {...}
        insertar(&arbol,nodo[0], nodo[1]);
       {...}
      }
      else
      {
       k = (int)(my_id + pow(2,i-1))%nproc;
       EnviarArbol(arbol, MPI_INT, k, tag, MPI_COMM_WORLD);
       {...}
       return;
      }
    }
    if (my_id == 0)
    {
     return arbol;
    }
}
```

En el resto de los ficheros aparecen las estructuras de datos como el árbol AVL, también las funciones encargadas de la lectura y escritura de ficheros de entrada y salida entre otras funcionalidades auxiliares del sistema. El archivo main.c es el punto de inicio del sistema y la interfaz principal a través del cual se hacen llegar los parámetros de entrada, los cuales son:

1. Modo de almacenamiento del modelo.
2. Formato de los datos de entrada.
3. Cantidad de nodos de procesamiento.

Una vez obtenidos los parámetros necesarios para iniciar el proceso de obtención del modelo se realiza la lectura de los datos de entrada (lista de registros) y se procede a la distribución por los diferentes nodos de procesamiento, concluida con esta tarea se da paso a la obtención del modelo por el algoritmo especificando la variante a emplear para almacenar el modelo (clásica, lista ordenada, árbol AVL), cada nodo obtiene el resultado de las comparaciones que le fueron asignadas y el algoritmo procede a realizar las sumas acumulativas donde el proceso con identificador cero es el encargado de recibir la suma final y devolver el modelo en el formato especificado.

3

Análisis de los Resultados

En este capítulo se presentan los resultados obtenidos por el sistema en su ejecución sobre un cluster Beowulf. Se realizan análisis comparativos en cuanto a los resultados obtenidos de forma teórica y empírica, concluyendo mediante una discusión sobre el cumplimiento de la hipótesis de investigación planteada.

Los resultados de todos los experimentos realizados fueron obtenidos a partir de datos de yacimientos lateríticos reales de la región norte del oriente cubano. La cardinalidad de las variables que determinan el modelo fueron fijadas en $|H| = 52, |P| = 9, |\alpha_H| = 17, |\alpha_V| = 7, |k| = 16$ que es la propuesta inicial de sus autores [23].

3.1 Evaluación Experimental del Algoritmo Secuencial

Algunas de las causas que influenciaron el bajo desempeño de las implementaciones propuestas en la sección de Soluciones Previas 1.3 fueron el alto costo en complejidad temporal y espacial que dificultaron la aplicación del modelo 1.2, y con la necesidad de nuevas soluciones y alternativas viables el Algoritmo 2.1.1 fue implementado en el lenguaje C sin los inconvenientes mencionados anteriormente.

Las pruebas fueron realizadas en una PC dedicada con procesador Intel (R) Core (TM) 2 Duo CPU E4500 2.20 GHz, memoria principal de 1 GB de RAM y sistema operativo Windows XP.

Los resultados para diferentes cantidades de muestra de entrada pueden observarse en la tabla 3.1, la cual muestra el tiempo de ejecución para $n = 28511$ obtenido por Ramírez et al. (2009), Peña (2007) [30] [27] y los alcanzados en la implementación secuencial propuesta. Se puede apreciar que los resultados fueron mejorados considerablemente.

Variante	Procesador	RAM	Lenguaje	Tiempo(minutos)
Peña 07	E6700 2.66 GHz	2 GB	Pascal	1056,67
Ramírez 09	E6700 2.66 GHz	2 GB	Pascal	489,67
Variante Secuencial AVL	E4500 2.20 GHz	1 GB	C	6.67
Variante Secuencial Clásica	E4500 2.20 GHz	1 GB	C	3.79
Variante Secuencial Lineal	E4500 2.20 GHz	1 GB	C	6.82

Tabla 3.1: Tiempo de ejecución para el Algoritmo Secuencial

Tomando como referencia la mejor de estas soluciones Ramírez 09 se puede concluir que la Variante Secuencial Clásica y la Variante Secuencial AVL fueron respectivamente 129 y 73 veces más rápida y la Variante Lineal fue 71 veces más rápida. Estos resultados no son una medida exacta de la relación de eficiencia entre las soluciones debido principalmente a que las ejecuciones no fueron realizadas en máquinas homogéneas tanto en hardware como en software, sin embargo sí permite apreciar las diferencias significativas de los resultados entre las anteriores investigaciones y la actual siguiendo sus tres variantes, Variantes Clásica, AVL y Lineal, las cuales fueron ejecutadas en este experimento en máquinas con menos prestaciones y aun así mostraron mejores resultados.

La implementación secuencial propuesta tuvo un comportamiento mejor que las anteriores como se mostró anteriormente, no obstante un análisis del comportamiento del tiempo de ejecución de esta en sus tres variantes con relación al tamaño de la muestra de

entrada, evidencia tiempos para nada pequeños, para el caso de n en el orden de los 512 mil registros el tiempo necesario para el modelado supera las 9 horas y en casos mayores del tamaño de entrada se alcanzan tiempos superiores a días como el caso para $n = 1024$ mil registros con un tiempo por encima de las 60 horas. En la tabla 3.2 se muestran algunos tiempos de ejecución en horas de las tres variantes para diferentes tamaños del problema.

Tamaño de Entrada (miles)	V. Sec. AVL	V. Sec. Clásica	V. Sec. Lineal
16	0.02	0.01	0.2
32	0.07	0.04	0.8
64	0.27	0.15	0.31
128	1.03	0.59	1.12
256	4.17	2.37	4.28
512	16.12	9.17	16.32
1024	64.55	36.73	65.23

Tabla 3.2: Tiempo de ejecución para el Algoritmo Secuencial (Horas)

Dada la frecuencia con que se necesita ejecutar este algoritmo y los tamaños del problema que generalmente se presentan durante el modelado de yacimientos lateríticos, puede apreciarse que, aun con las mejoras logradas, los tiempos son considerablemente altos.

3.2 Evaluación Experimental del Algoritmo Paralelo

Los resultados alcanzados mediante una aproximación paralela al problema serán abordados en esta sección. El Algoritmo 2.2.1 se implementó, en sus tres variantes, mediante el lenguaje C usando la librería MPI, los experimentos fueron realizadas en un cluster Beowulf de 8 PC con procesador Intel (R) Core (TM) 2 Duo CPU E4500 2.20 GHz y memoria principal de 1 GB de RAM con sistema operativo Linux Debian Lenny 5.0 y una red a 100 Mbps.

El tiempo paralelo del algoritmo, el *Speed-up* y la *eficiencia* están determinados por expresiones vistas en la sección 2.2.1 y las mismas dependen del tamaño de la muestra, de la implementación propia del algoritmo y del hardware en que se ejecute, lo cual determinará los parámetros tiempo para realizar una comparación c, número de procesadores p, tiempo de envío de una palabra t y el tiempo de latencia β.

Los experimentos realizados permitieron obtener el tiempo de comparación para las variantes y serán mostrados en esta sección, esto es debido a que calcular el tiempo paralelo

según la ecuación 2.1 sería poco exacto pues su valor sólo es una cota superior que se aleja del real al aumentar la dispersión de hipercubo ya que no todas las comparaciones realizadas aportan modificaciones.

La gráfica 3.1 muestra el tiempo de ejecución del algoritmo paralelo (Variante Clásica) en función del tamaño del problema y el número de procesadores. En la misma puede evidenciarse cómo a medida que el tamaño del problema aumenta, el tiempo para resolverlo lo hace con un factor de crecimiento de orden cuadrático tal como se expresa mediante la ecuación del tiempo paralelo 2.1 en el análisis teórico realizado al algoritmo.

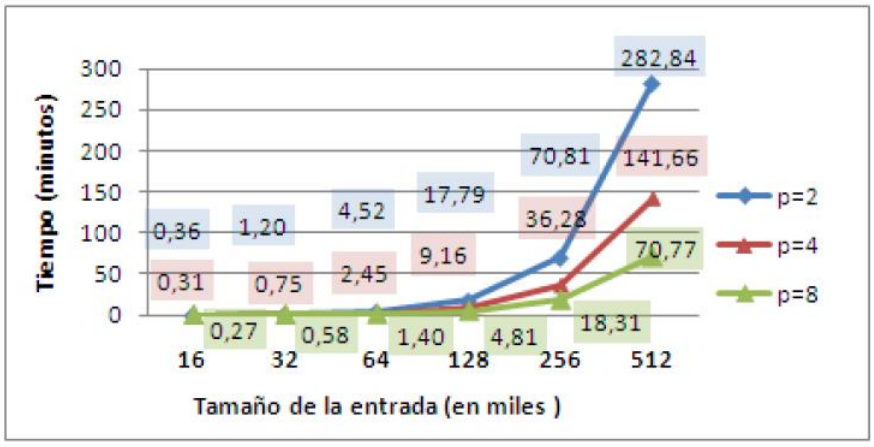

Figura 3.1: Tiempo de ejecución de la Variante Clásica usando 2, 4 y 8 procesadores

La gráfica 3.3 muestra el tiempo de ejecución del algoritmo paralelo en sus tres variantes en función del tamaño del problema para dos procesadores. Y en la gráfica 3.4 y 3.5 para 4 y 8 procesadores respectivamente.

En cada una de las gráficas obtenidas del análisis de algunos de los parámetros que rigen la ejecución del algoritmo paralelo, 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 de su misma variante. Si se realiza un análisis comparativo de las variantes paralelas con el algoritmo secuencial en la gráfica 3.5 puede notarse cómo el algoritmo paralelo usando 8 procesadores reduce considerablemente los tiempos respecto al algoritmo secuencial, si se considera el límite teórico p.

Se puede notar además que efectivamente la variante de la matriz completamente almacenada en memoria es la más rápida y la que representa 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 mayor 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 3.2 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 con el empleo de 6 procesadores y podemos apreciar que la diferencia de tiempo es muy significativa. Lo anterior se debe a la diferencia entre la complejidad temporal de ambas a la hora de insertar un elemento en sus respectivas estructuras de datos, resultando ser, $O(n)$ para el Vector Ordenado y $O(\log n)$ para el árbol AVL, durante las pruebas se observó que para dicha precisión del modelo el número de inserciones estuvo alrededor de un 97 % del total de las operaciones posibles a realizar provocando el aumento del consumo de tiempo.

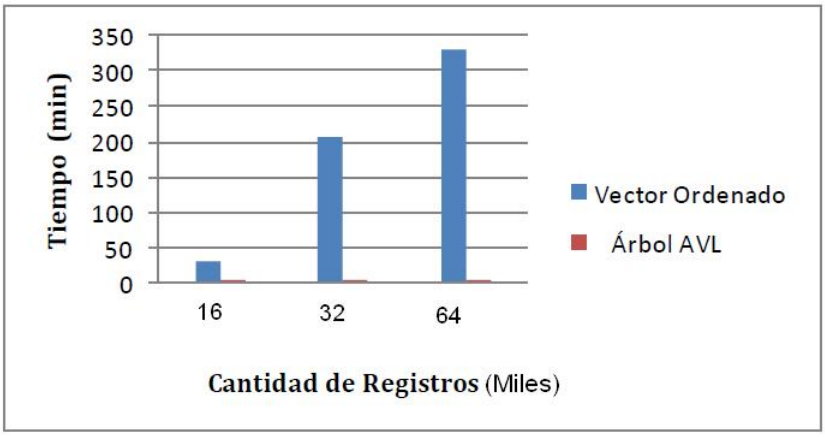

Figura 3.2: Tiempo de ejecución de la Var. AVL y Vec. Ordenado con $|\alpha_H| = 64, |\alpha_V| = 50$.

Tal como se demostró en la sección 2.3 los tiempos de la Variante AVL y Variante Lineal o Vector Ordenado son mayores que los requeridos por la Variante Clásica, para $n = 512$ mil y empleando la Variante AVL se consumió 2 horas, 26 minutos y 6 segundos al usar 8 procesadores y 4 horas, 49 minutos y 12 segundos al emplear 4 procesadores; ambos mayores a los vistos en la Variante Clásica donde para el problema de tamaño $n = 512$ mil el tiempo de ejecución del algoritmo usando 8 procesadores fue de 1 hora, 10 minutos y 5 segundos.

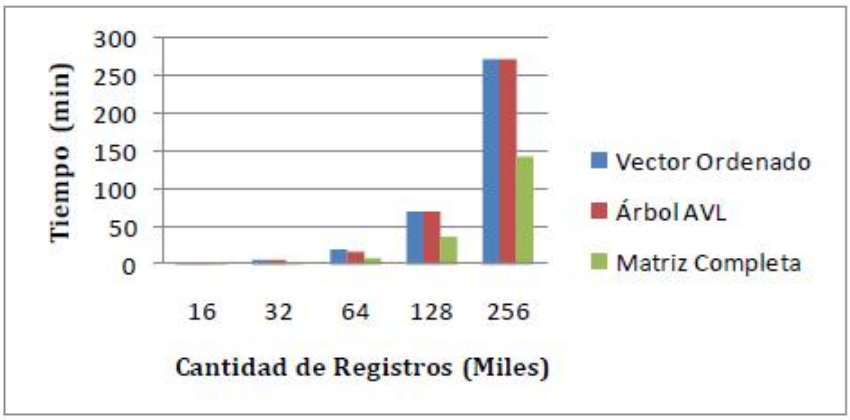

Figura 3.3: Tiempo de ejecución para el Algoritmo Paralelo con p=2 procesadores

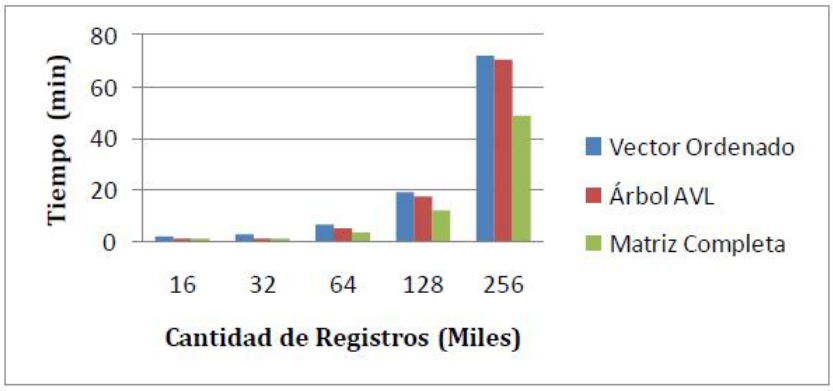

Figura 3.4: Tiempo de ejecución para el Algoritmo Paralelo con p=4 procesadores

3.2.1 Ganancia de Velocidad

Al realizar un análisis más detallado de la ganancia de velocidad mostrada por el algoritmo se analiza la aceleración que alcanza el algoritmo mientras se va aumentando el tamaño de entrada, es decir, qué tanto más rápido es en relación con el mejor tiempo secuencial que se obtuvo al ejecutar el algoritmo para un mismo tamaño del problema.

La gráfica 3.6 muestra como la Variante Clásica alcanza mejores resultados que la Variante AVL y Lineal, tendiendo a un Speed-Up óptimo a medida que aumenta el tamaño del problema, aunque la Variante AVL y Lineal también crece en prestaciones bajo las mismas condiciones.

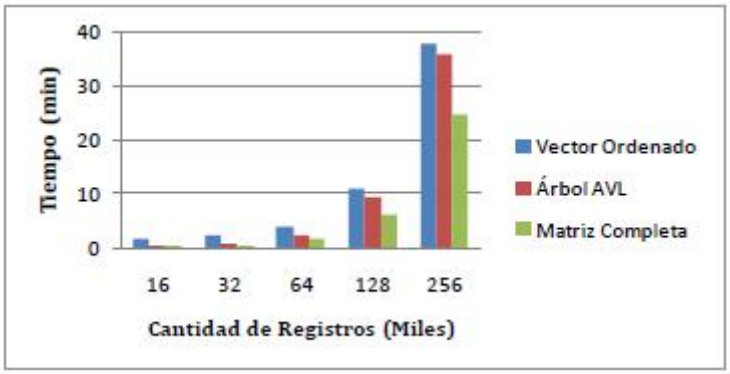

Figura 3.5: Tiempo de ejecución para el Algoritmo Paralelo con p=8 procesadores

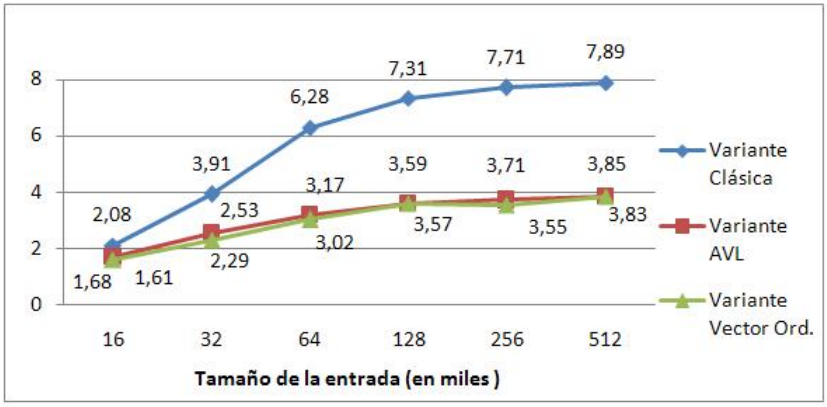

Figura 3.6: Speed-up por variante usando 8 procesadores

El Speed-up de la Variante Clásica se comportó como promedio 1.8 veces mayor que el mostrado por la Variante AVL. Y por otro lado la Variante AVL se mostró como promedio 3,09 veces más rápida que el algoritmo secuencial, logrando bajar los tiempos desde 9 horas, 30 minutos y 19 segundos a 2 horas, 26 minutos y 6 segundos. La mayor ganancia de velocidad estuvo dada para $n = 512$ mil por la Variante Clásica, donde se disminuyó el tiempo desde 9 horas, 30 minutos y 19 segundos a 1 hora, 18 minutos 5 segundos.

La ganancia de velocidad del algoritmo en la Variante Clásica en función del tamaño del problema y el número de procesadores se refleja en las gráficas 3.7 y 3.8.

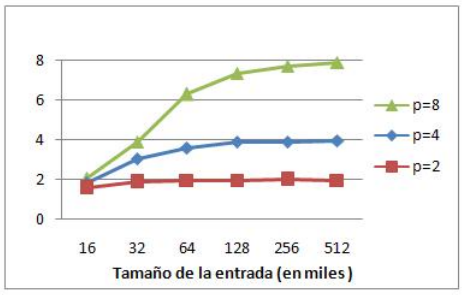

Figura 3.7: Speed-up en función de n

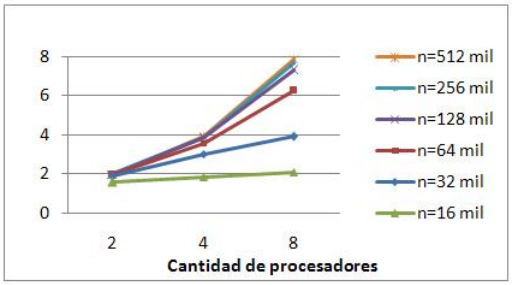

Figura 3.8: Speed-up en función de p

Independientemente de la cantidad de procesadores que se usen, el Speed-up del algoritmo tiende a su valor óptimo a medida que el tamaño del problema crece.

La ganancia de velocidad del algoritmo en la Variante AVL y Vector Ordenado en función del tamaño del problema se refleja en las gráfica 3.9, estas variantes también crece en prestaciones a medida que el tamaño del problema aumenta, a pesar de no tener un valor óptimo sino aproximadamente p/2.

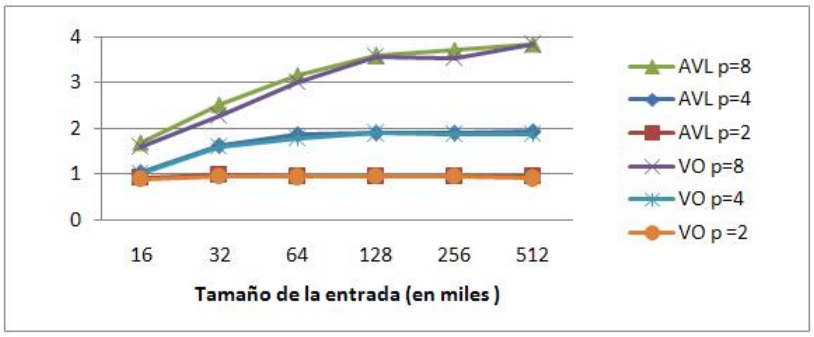

Figura 3.9: Speed-up en función de n

3.2.2 Análisis de la Eficiencia

La eficiencia usando 8 procesadores para las tres variantes del algoritmo se muestra en la gráfica 3.10 en la cual queda evidenciado el comportamiento que sigue este parámetro a medida que el tamaño del problema aumenta es la misma que la mostrada anteriormente por el Speed-up, se tiende a su valor óptimo en Variante Clásica a medida que el tamaño del problema crece y la Variante AVL y Lineal también mejora sus prestaciones bajo estas condiciones.

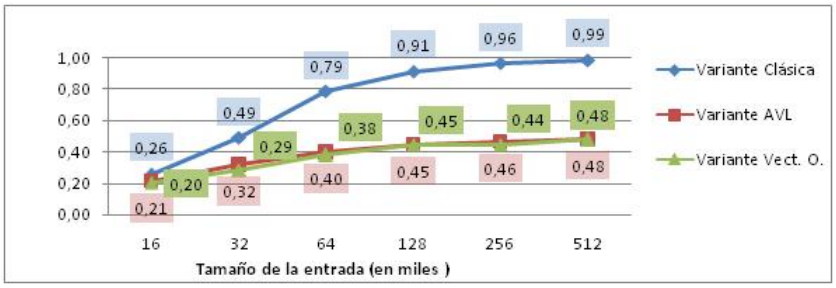

Figura 3.10: Eficiencia usando 8 procesadores

En la gráfica 3.11 se muestra la eficiencia del algoritmo en la Variante Clásica en función del tamaño del problema y puede apreciarse como la eficiencia aumenta con el crecimiento de n.

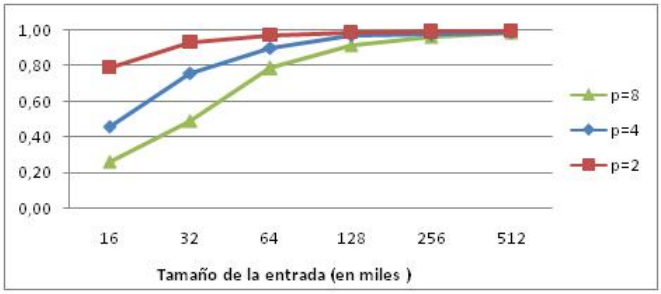

Figura 3.11: Eficiencia en función de n

En la gráfica 3.12 y 3.13 se muestra la eficiencia del algoritmo en la Variante AVL y Vector Ordenado respectivamente en función del tamaño del problema y puede apreciarse como la eficiencia aumenta con el crecimiento de n aun cuando no tiende a un valor óptimo si existe un crecimiento de este parámetro.

Escalabilidad

Según la definición de escalabilidad vista en la sección 1.4.3 un sistema es escalable si es posible obtener incrementalmente valores del tamaño del problema w definido como la complejidad del algoritmo que lo soluciona para los cuales la eficiencia crece o al menos se mantiene constante al aumentar p. Por lo que la escalabilidad de un sistema es una medida de su capacidad para incrementar el Speed-up en proporción al número de procesadores,

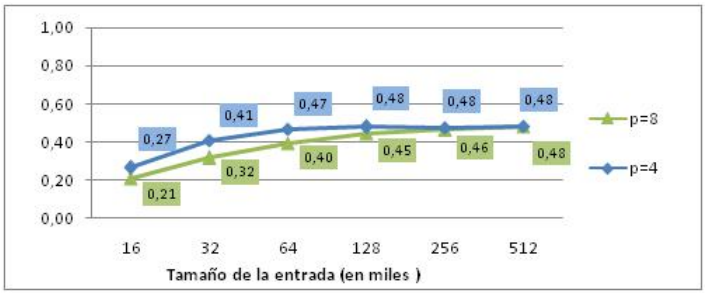

Figura 3.12: Eficiencia en función de n para AVL

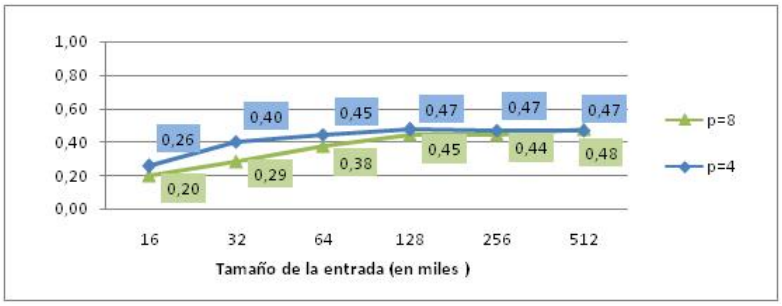

Figura 3.13: Eficiencia en función de n para Vect. Ordenado

reflejando la capacidad del mismo de utilizar de forma efectiva nuevos recursos que les sean incorporandos.

Empleando el Speed-up escalado se puede estudiar la escalabilidad del algoritmo paralelo, si su valor es cercana al número de procesadores entonces el algoritmo paralelo puede considerarse escalable. En la tabla 3.3 se muestran para la Variante Clásica los resultados de w, el Speed-up escalado y la eficiencia del sistema bajo las condiciones descritas anteriormente.

p	n(miles)	$w = n^2/2$	Speed-up Escalado	Eficiencia
2	256	$3{,}3 * 10^{10}$	1.99	0.997
4	362	$6{,}6 * 10^{10}$	3.96	0.989
8	512	$1{,}3 * 10^{11}$	7.89	0.986

Tabla 3.3: Escalabilidad

Como puede observarse al escalar el Speed-up la eficiencia puede considerarse constate, por lo que se concluye que el algoritmo tiene una buena escalabilidad.

3.3 Consideraciones del consumo de Memoria

El objetivo principal del modelado es obtener un modelo lo más representativo posible de la realidad del yacimiento, el cual está representado por un hipercubo como se ha venido viendo en secciones anteriores, en esta investigación se plantean algunas consideraciones sobre la idoneidad del uso de diferentes representaciones de este.

Como se señalaba en la sección 2.3 el espacio requerido para representar el hipercubo siguiendo la Variante Clásica es constante y para los valores definidos para los parámetros considerados en las pruebas y referidos al inicio de este capítulo la memoria necesaria fue de 54,387 MB.

Los experimentos mostraron un nivel de dispersión importante, cuyo mayor grado fue de un 14,3 % esto provocó que al emplear la Variante AVL y Vector Ordenado disminuyera significativamente el total de memoria necesaria durante el modelado alcanzándose ganancias en cuanto al consumo de la misma, y como se evidenció en la sección 2.3 el uso de estas Variantes AVL y Vector Ordenado son más óptimas que la Clásica siempre que los valores de dispersión sean menor a un 25 % y 50 % asociado respectivamente a cada una.

En la tabla 3.4 se muestra el consumo de memoria para el modelado empleando las tres variantes de representación del modelo para diferentes grados de dispersión y para los valores de cardinalidad de la dimensión del sector angular horizontal y vertical 17 y 7 respectivamente, estos resultados fueron obtenidos durante un grupo de pruebas realizadas a 3 yacimientos.

	Yac-1	Yac-2	Yac-3
Índice de Dispersión	2.45	13.86	14.05
Variante Clásica	54,39	54,39	54,39
Árbol AVL	4.57	25.84	26.20
Vector Ordenado	2.29	12.92	13.10

Tabla 3.4: Consumo de Memoria en MB con $|H| = 52, |P| = 9, |\alpha_H| = 17, |\alpha_V| = 7, |k| = 16$

En la Tabla 3.5 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, con el objetivo de aumentar la precisión del modelo y describir de forma más exacta la dirección de la dependencia entre clases, el gasto de memoria de la variante que almacena la matriz completa es excesivamente alto, llegando casi a los 1.5 Gb. 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 mucho menor, siendo la más ahorrativa la del Vector Ordenado.

	Yac-1	Yac-2	Yac-3
Índice de Dispersión	0.70	5.21	3.34
Variante Clásica	1462.5	1462.5	1462.5
Árbol AVL	41.05	304.65	195.60
Vector Ordenado	20.52	152.32	97.80

Tabla 3.5: Consumo de Memoria en MB con $|H| = 52, |P| = 9, |\alpha_H| = 64, |\alpha_V| = 50, |k| = 16$

De los resultados mostrados en la tabla anterior se puede determinar que la Variante Vector Ordenado utiliza como promedio la mitad de la memoria empleada por Variante AVL y esta última hace uso de memoria como promedio 15 veces menos que la Variante Clásica. De este análisis se llega a la conclusión que la variante más rápida es la que utiliza como estructura de datos para representar el modelo la Variante Clásica, 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. Teniendo en cuenta además que la Variante que emplea el Vector Ordenado hace el uso más eficiente de la memoria con relación al resto de las variantes pero para altos niveles de precisión aun cuando disminuye la dispersión del modelo aumenta el número de inserciones y esto trae consigo que aumente la complejidad temporal y con ella el consumo de tiempo.

3.4 Valoraciones finales

Tanto los resultados analíticos como prácticos demostraron que el algoritmo paralelo es aproximadamente p veces más rápido, siguiendo la Variante Clásica, que su versión secuencial siempre que se use, para su ejecución, un sistema paralelo dedicado con p procesadores.

Para disminuir el consumo de memoria se propusieron tres estructura de datos que permitieron durante las pruebas realizadas a partir de yacimiento lateríticos reales, un decremento importante en el consumo de memoria según se pudo observar en la sección anterior llegando a un ahorro de memoria, como promedio de un 70 a un 97 %, siempre condicionado por la dispersión y precisión del modelo.

El mayor valor de dispersión observado en los experimentos realizados fue de un $14,5\,\%$. Siempre que el nivel de dispersión del hipercubo fuera inferior a un $25\,\%$ se obtendrán

ganancias en cuanto a memoria con el empleo de la Variante AVL, es la variante más equilibrada en cuanto a consumo de memoria y tiempo.

La Variante Vector Ordenado hace un uso más eficiente de la memoria con relación al resto de las variantes pudiendo ser empleada siempre que el nivel de dispersión del hipercubo fuera inferior a un 50 % pero con el inconveniente de tener la mayor complejidad temporal y para altos niveles de precisión del modelo presenta una marcada diferencia por encima en el consumo de tiempo.

En muchos problemas el ahorro de memoria implica detrimento del parámetro tiempo y en otros, mejorar la precisión de la resolución conlleva un requerimiento mayor de memoria. En el problema que se aborda ambas premisas están presentes, por ejemplo, el algoritmo paralelo siguiendo la Variante AVL tiene una reducción en el consumo de memoria y fue $p/2$ veces más rápido que el algoritmo secuencial, sin embargo, fue aproximadamente 2 veces más lento que el algoritmo paralelo en su Variante Clásica. Este hecho permite concluir que aunque la ganancia de velocidad se compromete es posible mejorar simultáneamente ambos parámetros.

Conclusiones

Es posible reducir considerablemente el tiempo de modelado respecto a los enfoques secuenciales, su valor tiende a su óptimo teórico a medida que aumenta el tamaño del problema, aunque depende de la dispersión del modelo y la estructura de datos empleada.

Las estructuras de datos propuestas para la representación del hipercubo de probabilidades condicionales disminuyen significativamente la memoria necesaria para realizar el modelado, diferencia que se acentúa cuando la precisión del modelo crece.

El sistema paralelo que se propone en la presente investigación con el fin de modelar yacimientos lateríticos satisface las necesidades planteadas por el Centro de Investigaciones del Níquel de Moa en cuanto a complejidad temporal y espacial lo que facilita su introducción en el proceso de Exploración del Níquel.

Recomendaciones

- Implementar el sistema para su ejecución sobre otras arquitecturas como en entornos con memoria compartida o híbridas, de tal forma que pueda estudiarse el desempeño que este alcanza al explotar las bondades de los procesadores multinúcleos.
- Implementar el sistema para que sea capaz de aprovechar las Unidades de Procesamiento Gráfico(GPU).
- Integrar el sistema con el paquete MPE(MPI Parallel Environment) con el objetivo de proporcionar a los usuarios un grupo de herramientas útiles para el análisis y seguimiento de su ejecución.
- Estudiar otras formas de representación del modelado, debiéndose explorar las posibilidades que ofrecen las bibliotecas del Álgebra Lineal para el trabajo con matrices dispersas.

Referencias Bibliográficas

[1] AAL Lobaina, RGP Almanza, J. L. (2000). Informatización de la minería en la industria cubana del níquel. In *biblioteca.ismm.edu.cu*, Moa, Holguín.

[2] Al. Geist, A. Beguelin, J. D. W. J. R. M. V. S. (1994). *Parallel Virtual Machine. A Users' Guide and Tutorial for Networked Parallel Computing*. Scientific and Engineering ComputationJanusz Kowalik.

[3] Allen, M. (1999). Do-it-yourself climate prediction. *Nature*, 401(6754):642–642.

[4] Almeida, F., Giménez, D., Mantas, J. M., and Vidal, A. M. (2008). *Introducción a la programación paralela*. Thomson Paraninfo.

[5] Anderson, D., Cobb, J., Korpela, E., Lebofsky, M., and Werthimer, D. (2002). SETI@ home: an experiment in public-resource computing. *Communications of the ACM*, 45(11):61.

[6] Arias, J., Pérez, M., and Campo, M. (2005). Determinacion de la continuidad de la mineralizacion del horizonte de serpentinitas duras (sd) en el yacimiento yamaniguey. In *I Convención Cubana de Ciencias de la Tierra, I Congreso de Minera (MIN3-6)*.

[7] Ariosa, J. D. (1977). *Curso de Yacimientos Minerales Metálicos tipo genético*. Pueblo y Educación, La Habana.

[8] Cuador, J. Q. (2002). *Estudios de Estimación y Simulación Geoestadística para la Caracterización de Parámetros Geólogo - Industriales en el Yacimiento Laterítico Punta Gorda*. Tesis doctoral.

[9] Dongarra, J., Foster, I., Fox, G., Gropp, W., Kennedy, K., Torczon, L., and White, A. (2003). *Sourcebook of parallel computing*. Morgan Kaufmann Publishers San Francisco, CA.

[10] EGS (1985). Sistema automatizado para el cálculo de reservas en yacimientos niquelíferos. Moa, Holguín.

[11] Flynn, M. (1972). Some computer organizations and their effectiveness. *IEEE Transactions on Computers*, 100:21.

[12] Foster, I. (1995). *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA. 527029.

[13] García, Y. (2010). Balance anual del grupo empresarial cubaníquel. Moa, Holguín.

[14] Gómez, O., Estévez, E., and Cuador, J. Q. (2005). Modelaje geológico y de recursos del yacimiento "pastelilloütilizando el krigeaje de indicadores. In *I Convención Cubana de Ciencias de la Tierra, VI Congreso de Geología (GEO 14-20)*.

[15] Grama, A., Gupta, A., Karypis, G., and Kumar, V. (1994). *Introduction to Parallel Computing. Design and Analysis of Algorithms*. The Benjamin/Cummings Pub. Company, Redwood City, California.

[16] Habata, S., Yokokawa, M., and Kitawaki, S. (2003). The earth simulator system. *NEC Research and Development*, 44(1):21–26.

[17] Lavaut, W. (1998). Tendencias geológicas del intemperismo de las rocas ultramáficas en cuba oriental. *Minería y Geología*, 15(1):9–16.

[18] Legrá, A. A. (1999). *Metodología para el pronóstico, planificación y control integral de la minería en yacimientos lateríticos*. Ph d, Instituto Minero-Metalúrgico de Moa.

[19] León, A. R. (2008). Programación de alto desempeño. Instituto Tecnológico de Veracruz.

[20] Martínez, A., Legrá, A. A., Ferrera, N., and Mena, L. (2003). Determinación de un modelo digital de la topografía original en el yacimiento punta gorda. *Revista Minería y Geología*, XVIII(3-4):103–119.

[21] Morrison, R. (2003). *Cluster Computing: Architectures, Operating Systems, Parallel Processing & Programming Languages*. Sydney University of Technology, Australia.

[22] ORG Regis, EC Martínez, H. C. (2003). *Computación Paralela*. dynamics.unam.edu.

[23] Peña, R. E. (2007a). Algoritmo de conteo para modelos markovianos en yacimientos lateríticos. In *COMPUMAT*, La Habana, Cuba.

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

[25] Peña, R. E., Labrada, A. S., Céspedes, J. G., and Rodríguez, L. M. (2009a). Multiprocesamiento con tecnología cliente-servidor para el modelado markoviano de yacimientos lateríticos. In *III Convención de Ciencias de la Tierra*, La Habana, Cuba.

[26] Peña, R. E. and Legrá, A. (2005). Nuevo enfoque al problema de la optimización de redes de exploración en yacimientos lateríticos. In *I Convención Cubana de Ciencias de la Tierra*, La Habana, Cuba.

[27] Peña, R. E., Matos, L., Ortiz, E., and Robles, V. (2007). Propuesta de clases patrones en yacimientos lateríticos ferro-niquelíferos. In *II Convención Cubana de Ciencias de la Tierra*, La Habana, Cuba.

[28] Peña, R. E., Ramírez, A., Broscat, L. Y., and Gorina, A. (2009b). Distribución equitativa de tareas en un algoritmo de conteo paralelizado sobre una red de área local. aplicación al modelado de yacimientos lateríticos. In *COMPUMAT*, La Habana, Cuba.

[29] Pimentel, H., Gómez, O., Gala, T., Estévez, E., and Cuador, J. Q. (2005). Evaluación geólogo-económica de las menas oxidadas del yacimiento de oro-cobre golden hill. In *I Convención Cubana de Ciencias de la Tierra, VI Congreso de Geología (GEO 13-7)*.

[30] Ramírez, A. M., Peña, R. E., and Broscat, L. Y. (2009). Mejora de un algoritmo de conteo para modelos markovianos en yacimientos lateríticos. In *COMPUMAT*, La Habana, Cuba.

[31] Rojas, A. (1994). *Principales Fases Minerales Portadoras de Níquel en los Horizontes Lateríticos Del Yacimiento Moa*. Tesis doctoral, ISMM.

[32] RT Codorníu, ML Vidal, M. L. (2000a). Algoritmo eficiente para procesos de estimaciÓn geoestadÍsticos. In *biblioteca.ismm.edu.cu*, Moa, Holguín.

[33] RT Codorníu, ML Vidal, M. L. (2000b). Sistema para el almacenamiento de la información geológica en empresas del níquel. In *biblioteca.ismm.edu.cu*, Moa, Holguín.

[34] Sterling, T., Becker, D., Savarese, D., Dorband, J., Ranawake, U., and Packer, C. (1995). BEOWULF: A parallel workstation for scientific computation. In *Proceedings of the 24th International Conference on Parallel Processing*.

[35] TOP500.Org (2010). Top 500 supercomputer sites.

[36] Trinchet, D. A. (2010). *Algoritmo paralelo para el modelado de yacimientos lateríticos.* Tesis maestria, Universidad de las Habana, Ciudad de La Habana, Cuba.

[37] Vera Sardinas, L. O. (2001). *Procedimiento para la determinación de las redes racionales de exploración de los yacimientos lateríticos de níquel y cobalto en la región de Moa*. Tesis doctoral, ISMM.

[38] Vera Yeste, A. (1979). *Introducción a los yacimientos de níquel cubanos.* ORBE, Ciudad de la Habana.

[39] Villavicencio, B. (2005). Una aplicación de redes neuronales artificiales en registros geofísicos de pozos. In *I Convención Cubana de Ciencias de la Tierra, VI Congreso de Geología (GEO 14-3).*

[40] Wilkinson, B. and Allen, M. (1998). *Parallel programming: techniques and applications using networked workstations and parallel computers.* Prentice Hall.

Anexo A
Otros parámetros del modelo

Discretización de la Dirección Horizontal

El índice α_H, que representa la dirección horizontal de la muestra destino c_j respecto a la muestra origen c_i, expresa la dirección en función de una de las 16 *direcciones* de la rosa náutica donde el ángulo 0 fija el Oeste y π el Este (ver figura 14).

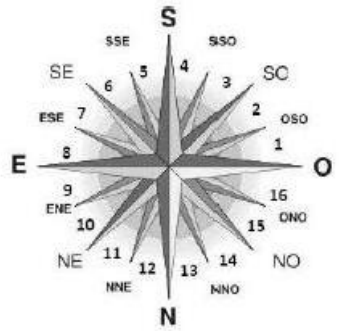

FIGURA 14: Discretización de la dirección horizontal

Primeramente se calcula el ángulo respecto a la dirección Oeste absoluta y luego se determina el sector angular de amplitud $\pi/8$ al cual pertenece discretizándolo mediante la función:

$$f(x) = \begin{cases} n+1 & \text{si } \exists n \in [0,15] \text{ tal que } \varphi \in [\frac{\pi n}{8}, \frac{\pi(n+1)}{8}] \\ 0 & \text{si } n \to \infty \end{cases}$$

Cálculo del parámetro profundidad

La profundidad (H) de una comparación se determina como el máximo de las profundidades de las muestras origen c_i y destino c_j a comparar.

Cálculo del parámetro Paso

El paso (P), que expresa la distancia entre las muestras a comparar, se define como la cantidad de pozos que separa ambas muestras dentro de la red de exploración, por lo que P puede determinarse mediante la Distancia de Chebyshev: $P = \text{máx}(|X_1|-|X_2|, |Y_1|-|Y_2|)$.

Anexo B
Implementación de algunas funciones usando MPI

Función para la distribución inicial de los registros a los nodos de procesamientos para que puedan realizar las comparaciones que les corresponden.

```
void DistribucionDatos_Nodos(Registro* records, int CantRegistros,
                             int my_id)
{
int val_reg[4];
int j =0;
for(j =0; j<CantRegistros ;j++)
 {
     if (my_id == 0)
     {
      val_reg[0] = records[j].posBloque;
      val_reg[1] = records[j].posPozo;
      val_reg[2] = records[j].posProfundidad;
      val_reg[3] = records[j].posCaracteristica;
     }
   MPI_Bcast(val_reg,4,MPI_INT,0,MPI_COMM_WORLD);
    records[j].posBloque = val_reg[0] ;
    records[j].posPozo  = val_reg[1];
    records[j].posProfundidad = val_reg[2];
    records[j].posCaracteristica = val_reg[3];
 }
}
```

Función para realizar la suma acumulativa de los Hipercubos, en este caso el modelo esta representado según la variante AVL.

```
void EnviarArbol(AVLTree *t,MPI_Datatype datatype, int dest,
                 int tag, MPI_Comm comm)
{
  int nodo[2];
  if(!t) return;
  if(t->key)
  {
   nodo[0] = t->key;
   nodo[1] = t->dato;
   MPI_Send(nodo,2,datatype,dest,tag,comm);
  }
  EnviarArbol(t->izq, MPI_INT, dest, tag, MPI_COMM_WORLD);
  EnviarArbol(t->der, MPI_INT, dest, tag, MPI_COMM_WORLD);
}
```

Lista de Símbolos y Abreviaturas

Abreviatura	Descripción	Definición
CEINNIQ	Centro de Investigaciones del Níquel depag. Moa	pag. x
ORENI	Proceso de Optimización de Redes de Exploración del Níquel	pag. x
MPT	Matriz de Probabilidades de Transición	pag. 4
CPU	Del Inglés Central Processing Unit, Unidad Central de Procesamiento	pag. 7
SPMD	Del Inglés Single Program Multiple Data, Programa Único que actúa sobre Múltiples Datos	pag. 11
SISD	Del Inglés Message Passing Interface, Interfaz de Paso de Mensajes	pag. 10
SIMD	Del Inglés Single Instruction stream, Multiple Data stream, Flujo simple de instrucciones que actúa sobre varios conjuntos de datos	pag. 10
MISD	Del Inglés Multiple Instruction stream, Single Data stream, Diferentes flujos de instrucciones que actúa sobre un único conjunto de datos	pag. 10
MIMD	Del Inglés Multiple Instruction stream, Multiple Data stream, Diferentes flujos de instrucciones que actúan sobre diferentes conjuntos de datos	pag. 10
MPI	Del Inglés Message Passing Interface, Interfaz de Paso de Mensajes	pag. 20
AVL	Árbol Binario de Búsqueda Balanceado ideado por los matemáticos rusos Adelson-Velskii y Landis	pag. 30

Índice de tablas