Universidad de las Ciencias Informáticas

Facultad 6

Algoritmo para la Interpolación Espacial de Krigeado Ordinario

Tesis presentada por Milenis Fernández Díaz
para obtener el grado de Máster en Informática Avanzada

Supervisada por:
Dr. José Quintín Cuador Gil
Dr. César Raúl García Jacas

2016

La Habana, Cuba

Dedicado a *mis padres Amparo y Félix,*
A mi querida abuela Rosa,
A mis hermanos Miguel Ángel y Ariadna,
A mi novio José Gabriel.

Agradezco *especialmente*
A mis tutores, José Cuador y César Raúl,
A mi amigo Eddy Dangel,
A Nayma, Eduanys y Raciel,
Al geólogo José Arias,
A mi novio José Gabriel.

Declaración jurada de autoría

Declaro por este medio que yo Milenis Fernández Díaz, con carnet de identidad 88112418656, soy el autor principal del trabajo final de maestría que lleva por título "Algoritmo para la Interpolación Espacial de Krigeado Ordinario", desarrollado como parte de la Maestría de Informática Avanzada y que autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo. Finalmente declaro que todo lo anteriormente expuesto se ajusta a la verdad y asumo total responsabilidad moral y jurídica que se derive de este juramento profesional. Y para que así conste, firmo la presente declaración jurada de autoría en La Habana a los ____ días del mes ____ del año 2016.

Firma del maestrante Firma de supervisores

Resumen

El cálculo de las reservas y recursos minerales es de capital importancia para las empresas geológico-mineras, tanto para la evaluación como para la planificación de la explotación de las reservas. Se puede realizar utilizando diversas técnicas, predominando los métodos geoestadísticos, los cuales tienen en cuenta las características de variabilidad y correlación espacial de los datos originales. La interpolación de Krigeado Ordinario es uno de los métodos geoestadísticos más frecuentemente usados para el cálculo de las reservas y recursos minerales. Su objetivo consiste en encontrar el Mejor Estimador Lineal Insesgado a partir de los datos disponibles, los cuales generalmente son insuficientes debido al costo de su obtención. Se caracteriza por costosas operaciones de álgebra lineal que repercuten en altos tiempos de ejecución, fundamentalmente la resolución de grandes sistemas de ecuaciones lineales. Con el objetivo de disminuir los tiempos asociados a la interpolación espacial de Krigeado Ordinario, se propuso un algoritmo basado en el uso de técnicas de programación paralela, así como métodos optimizados de búsqueda espacial; que permita resolver los problemas de estimación en tiempos razonables, fundamentalmente en el campo de las Geociencias. Este algoritmo fue implementado usando C++11 como lenguaje de programación, OpenMP 4.8.2 como biblioteca de programación paralela en memoria compartida, y Atlas CLapack como biblioteca de álgebra lineal optimizada para los cálculos matriciales. El algoritmo propuesto ofrece menores tiempos en los procesos de estimación de reservas de minerales útiles, a partir del aprovechamiento eficiente de los recursos computacionales.

Palabras clave: algoritmo paralelo, interpolación espacial, Krigeado Ordinario, métodos geoestadísticos

Índice general

Introducción

Los recursos minerales son la base de la sociedad moderna, y su demanda ha aumentado aceleradamente con el desarrollo tecnológico. Los recursos minerales son finitos y, en su mayoría, no renovables, haciendo que la estimación de sus depósitos sea un elemento primordial, tanto para evaluar las reservas como para garantizar la explotación de los yacimientos en condiciones de rentabilidad económica. La estimación de recursos minerales aporta los elementos necesarios para evaluar la factibilidad económica de los proyectos mineros, así como garantizar planificaciones adecuadas para estos.

El cálculo de las reservas y recursos minerales es de capital importancia para las empresas geológico-mineras. Constituye una operación de alta responsabilidad que determina el valor comercial o industrial de un yacimiento (Cuador, 2002). Proporciona una cuantificación formal de las materias primas minerales, la cual se conoce como inventario de mineral. Las estimaciones se pueden realizar a escala global cuando se quiere estimar la ley media de todo el yacimiento, o a escala local, para estimar la ley media de unidades o bloques con el fin de encontrar zonas ricas o pobres en recursos minerales (Alfaro, 2007).

La estimación de recursos minerales se realiza a partir de los datos obtenidos durante las campañas de prospección y exploración, a través de perforaciones realizadas sobre la corteza terrestre. Usualmente los datos disponibles no cubren completamente el dominio de interés debido a que su adquisición resulta difícil o costosa. Los cálculos se basan en un número limitado de muestras, sujetas a errores respecto a los valores reales; lo cual conlleva a la necesidad de clasificar los recursos y reservas de minerales útiles indicando los riesgos asociados a la estimación (Estévez, 2002).

El cálculo de los recursos y reservas minerales se puede realizar utilizando diversas técnicas con distintos niveles complejidad, agrupados en métodos clásicos o tradicionales (como la media aritmética, la media ponderada, el método de los bloques geológicos, el método de los bloques en explotación, el método de los polígonos, el método de las isolíneas, el método de los perfiles, la triangulación); y métodos modernos basados en procedimientos matemáticos de interpolación espacial de los datos (como el inverso de la distancia y el Krigeado, en todas

sus variantes).

Los métodos clásicos, empleados desde los inicios de la minería se basan en criterios de interpretación de las variables entre dos puntos contiguos de muestreo (Estévez, 2002). A pesar de que los métodos tradicionales son fáciles de calcular, muchos autores no recomiendan su utilización. Estos métodos aunque permiten cuantificar las reservas a escala global, no se consideran adecuados para la caracterización local de las reservas (Cuador, 2002). Además, no consideran la estructura del fenómeno mineralizado, entiéndase estructura como la continuidad de las leyes y las direcciones en las cuales se favorece la variación de estas. Se plantea como una de sus limitaciones el hecho de que no proporcionan el error asociado a las estimaciones (Alfaro, 2007).

En su tesis doctoral (Cuador, 2002) considera que la utilización de métodos tradicionales para la estimación de los reservas nacionales, arroja resultados inexactos que repercuten negativamente en la explotación racional de los recursos minerales. La complejidad de los yacimientos en cuanto a la composición química y mineralógica 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 (Peña y Legrá, 2005). La práctica actual en la actividad minera, y en la geociencias de forma general, exige de métodos modernos de estimación.

La necesidad de obtener mayor precisión en las estimaciones de reservas condujo al surgimiento y desarrollo de la Geoestadística. Danie Gerhardus Krige junto a Herbert Sichel llevaron a cabo los primeros pasos de esta ciencia (1951), al explotar la correlación espacial para hacer predicciones en la evaluación de reservas de las minas de oro en Sudáfrica (Díaz, 2002). Estas técnicas fueron formalizadas por el matemático francés George Matheron (1963) junto a su grupo de la Escuela de Minas de París, quién sentó las bases de lo que se conoce como la Teoría de la Variable Regionalizada (Matheron y Kleingeld, 1987).

La Geoestadística fue definida por Matheron como "la aplicación del formalismo de las funciones aleatorias al reconocimiento y estimación de fenómenos naturales"(Matheron, 1962), fundamentalmente en la minería. Según (Díaz, 2002) la Geoestadística es una "rama de la estadística aplicada que se especializa en el análisis y la modelación de la variabilidad espacial en ciencias de la tierra". De forma abreviada la Geoestadística, según (Le, 2005), puede definirse como la "ciencia que estudia las variables regionalizadas". Entiéndase por variables regionalizadas, "variables numéricas distribuidas en el espacio" (Díaz, 2002), es decir, "funciones que representan la variación en el espacio de una cierta magnitud asociada a un fenómeno natural" (Alfaro, 2007).

Aunque el surgimiento de la Geoestadística tuvo lugar en el campo de la minería, ha sido

ampliamente aplicada a muchas áreas de la ciencia, tales como: la hidrología (Goovaerts, 2000), ciencias ambientales (Jeffrey et al., 2001), detección de sismos (Richard et al., 1994), ciencias atmosféricas (Vasan y Murtugudde, 2010), agronomía y meteorología (Webster y Oliver, 2007), entre otras. Su importancia radica en la descripción de la continuidad espacial de las variables y la estimación de valores muy cercanos a los reales en puntos desconocidos, sustentado en técnicas de interpolación espacial. Estas técnicas proporcionan a la Geoestadística el análisis y simulación de una muestra de datos, así como su comportamiento en el espacio e influencia en otros puntos (Gomariz, 2013).

Los métodos geoestadísticos ayudan a los investigadores en la construcción de modelos confiables considerando la variabilidad y correlación espacial del fenómeno en estudio (Allombert et al., 2014), así como la estimación de variables ambientales en localizaciones no muestreadas (Gutiérrez de Ravé et al., 2014)). Estos métodos ofrecen mejores resultados, ya que parten del concepto intuitivo de que mientras más cerca estén dos puntos sobre la superficie terreste, menor será la variación del atributo medido entre ellos (Deraisme y De Fouquet, 1996). Los métodos geoestadísticos han alcanzado una alta difusión, puesto que la complejidad de su aplicación se ha reducido con el uso de la computación. La interpolación de Krigeado es uno de los métodos geoestadísticos más frecuentemente usados para la realización de análisis espaciales (Kleijnen, 2009).

La estimación de variables aleatorias en puntos no muestreados se conoce como Krigeado (*Kriging* en inglés), en honor a Danie G. Krige por su contribución en la aplicación del análisis de regresión entre muestras (Martínez y Ramirez, 2005). El Krigeado constituye el estimador utilizado en los métodos geoestadísticos; su objetivo es encontrar el Mejor Estimador Lineal Insesgado[1] a partir de los datos disponibles (Chica, 1997), los cuales generalmente son insuficientes debido al costo de su obtención. A diferencia de otros métodos de interpolación, como por ejemplo el inverso de la distancia, el Krigeado tiene en cuenta las características de variabilidad y correlación espacial del fenómeno que se estudia.

El Krigeado constituye "una colección de técnicas generalizadas de regresión lineal para minimizar una varianza[2] de estimación definida de un modelo a priori de covarianza[3]" (Olea, 1991). El Krigeado tiene muchas variantes según los grados de estacionariedad de la función aleatoria que representa el fenómeno regionalizado; pero la más utilizada es el Krigeado Ordinario. En las últimas décadas, estos métodos han evolucionado hacia técnicas no lineales basadas en transformaciones no lineales de los grados de continuidad. Más recientemente se

[1] Un estimador se considera insesgado cuando la esperanza (el valor esperado) es igual al verdadero valor del parámetro a estimar.

[2] Medida de dispersión de los datos con respecto a la media.

[3] Grado de variación conjunta de dos variables aleatorias.

han introducido métodos de simulación condicional en la estimación de recursos minerales, pero la complejidad y la intensidad de los cálculos han dificultado la comprensión y aceptación de estos métodos (Glacken y Snowden, 2001).

La cantidad de cálculos que implica el uso de estos métodos hace imprescindible la utilización de medios automatizados para el proceso de estimación. En el ámbito internacional existen diversos programas informáticos profesionales que aplican los métodos geoestadísticos para la estimación de las reservas, entre estos se puede mencionar: DATAMINE, GEMCOM, VULCAN, MICROMINE, GEOPACK, SURPAC, GEOSTAT, MICROLYNX, MINEMAP (Cuador, 2002). En el país se han introducido pocos sistemas mineros, debido fundamentalmente a los altos costos por conceptos de adquisición, actualización y capacitación; y también por causa del bloqueo económico impuesto por el gobierno norteamericano.

El Krigeado involucra operaciones de diferente índole, dígase: operaciones matriciales (inversión, multiplicación, sistemas de ecuaciones lineales), geométricas (distancia), estadísticas y de procesamiento de datos (Gutiérrez de Ravé et al., 2014). Se caracteriza por costosas operaciones de algebra lineal que repercuten en altos tiempos de ejecución. El análisis computacional del algoritmo realizado por (Pesquer y Pons, 2011) mostró que la resolución del sistema de ecuaciones lineales consume la mayor parte del tiempo de ejecución, aproximadamente de un 96.0 % a un 99.8 %. En este sentido, se conoce que la complejidad temporal de la resolución de un sistema de ecuaciones lineales usando la eliminación gaussiana regular es de N^3 (Allombert et al., 2014).

La complejidad de los cálculos matemáticos utilizados en la interpolación de Krigeado, fundamentalmente la resolución de grandes sistemas de ecuaciones, tiene un alto costo computacional confirmado por varios autores: (Kerry y Hawick, 1998) (Sullivan y Unwin, 2002) (Lloyd, 2006) (Vasan y Murtugudde, 2010) (Pesquer y Pons, 2011). Se plantea que el algoritmo por cada punto de interés tiene una complejidad cúbica, lo que conduce a una complejidad de $O(MN^3)$, siendo N el número de muestras y M el número de bloques (Vasan y Murtugudde, 2010). Cuando $M \approx N$ la complejidad computacional puede considerarse $O(N^4)$ lo cual no es favorable cuando se trabaja con yacimientos complejos que precisan de grandes volúmenes de datos.

Experimentos realizados con diferentes propósitos evidencian la necesidad de rapidez en las operaciones de estimación de recursos minerales, y en otras aplicaciones de interpolación espacial, en general. En una estación de trabajo con procesador Intel Core2 Quad Core con una frecuencia de 3 GHz y 4 GB de memoria RAM, para un juego de datos compuesto por 8 millones de puntos, el tiempo de ejecución fue de aproximadamente dos días (Vasan y Murtugudde, 2010). Otro experimento realizado sobre 486,400 pixeles en un procesador

Pentium IV de 2.6 GHz con 768 MB de RAM, tardó 11 h 26 m 12 s (Pesquer y Pons, 2011).

De lo expresado anteriormente se concluyen los siguientes aspectos:

- La interpolación de Krigeado Ordinario en máquinas convencionales no es viable cuando se trabaja con yacimientos complejos que precisan de grandes cantidades de datos, debido a que se intensifican los cálculos, y los tiempos de ejecución se elevan.
- La complejidad de los cálculos matemáticos utilizados en la interpolación de Krigeado Ordinario, fundamentalmente la resolución de grandes sistemas de ecuaciones, repercute en altos tiempos de ejecución que retardan el proceso de estimación de recursos minerales.
- Las operaciones de estimación de reservas minerales útiles a través de métodos geoestadísticos se caracterizan por el procesamiento intensivo de cálculos, donde la velocidad de ejecución depende directamente del número de puntos a interpolar, así como el tamaño de la vecindad definida.

La situación antes expuesta origina el siguiente **problema científico**: ¿Cómo disminuir los tiempos de ejecución del algoritmo de interpolación de Krigeado Ordinario en los procesos de estimación de recursos minerales?

La investigación tiene como **objetivo general**: Desarrollar un algoritmo de interpolación de Krigeado Ordinario, basado en el uso de técnicas de programación paralela, así como métodos optimizados de búsqueda espacial, que permita realizar la estimación de recursos minerales en tiempos razonables.

Para dar cumplimiento al objetivo general se trazaron los siguientes **objetivos específicos**:

- Elaborar el marco teórico-referencial de la investigación relacionado con la interpolación de Krigeado Ordinario aplicada a la estimación de recursos minerales.
- Diagnosticar las limitaciones del algoritmo de interpolación de Krigeado Ordinario en cuanto al tiempo de ejecución.
- Identificar las oportunidades de optimización y paralelización en el algoritmo de interpolación de Krigeado Ordinario.
- Implementar el algoritmo de interpolación de Krigeado Ordinario aplicando técnicas de programación paralela y métodos optimizados de búsqueda espacial.
- Evaluar experimentalmente el algoritmo de interpolación de Krigeado Ordinario a partir de su aplicación a conjuntos de datos que simulan yacimientos de gran tamaño.

El análisis de los elementos antes expuestos conduce a plantear como **hipótesis investigativa**: El algoritmo de interpolación de Krigeado Ordinario, basado en el uso de técnicas de programación paralela, así como métodos optimizados de búsqueda espacial, permitirá dis-

minuir los tiempos de ejecución del Krigeado Ordinario en los procesos de estimación de recursos minerales.

Se identifica como **variable independiente** el algoritmo de interpolación de Krigeado Ordinario, y como **variable dependiente**: el tiempo de ejecución del algoritmo de Krigeado Ordinario. Entiéndase como algoritmo el conjunto finito de pasos utilizados por una computadora para resolver un problema, en este caso, la interpolación de Krigeado Ordinario. Entiéndase tiempo de ejecución como el tiempo requerido por el algoritmo para encontrar la solución para un tamaño de entrada y prestaciones de hardware específicos.

Como **variables ajenas** se identifican: el tamaño de los de datos de entrada, las capacidades de hardware instaladas (dígase: memoria, reloj, procesador) y el sistema operativo (dígase: las características multitarea y la carga del sistema). La operacionalización de estas variables se encuentra definida en el Apéndice A.

Para el desarrollo de la investigación se utilizaron los siguientes **métodos científicos**:

Métodos teóricos

- Analítico-Sintético: Para estudiar las características de la interpolación de Krigeado Ordinario aplicada a la estimación de recursos minerales, y posteriormente extraer los elementos fundamentales para la investigación.
- Inductivo-Deductivo: Para la deducción de conclusiones a partir del estudio realizado sobre el proceso de interpolación de Krigeado Ordinario, así como determinar cuáles son las alternativas viables a incorporar en la presente investigación.
- Hipotético-Deductivo: Para la formulación de la hipótesis a partir del sistema de conocimientos y luego llevar a cabo su comprobación experimental.
- Histórico-lógico: Para el análisis del surgimiento y evolución de los métodos de interpolación de Krigeado, el análisis de las soluciones previas, así como las tendencias actuales en la implementación de algoritmos con alta complejidad temporal.
- Modelación: Para la representación simplificada del algoritmo a través de la modelación de sus pasos, entradas y salidas.

Métodos empíricos

- Análisis documental: Para la revisión de la literatura especializada en el tema, así como la extracción de los referentes teóricos que sustentan la investigación.
- Medición: Para la medición de los tiempos de ejecución de diferentes corridas del algoritmo y luego establecer comparaciones entre los resultados obtenidos, utilizando el reloj de la computadora como instrumento de medición.
- Experimento controlado: Se realizó la evaluación experimental del algoritmo variando el tamaño de los datos y el número de procesadores, con el objetivo de demostrar la reduc-

ción de los tiempos de ejecución del algoritmo a partir de su implementación utilizando técnicas de computación paralela y métodos optimizados de búsqueda espacial.
- Estadísticos descriptivos: Usados para describir las mediciones realizadas a los tiempos de respuesta del algoritmo, mediante estadígrafos, tablas y gráficos que facilitan el análisis de estos datos, permitiendo arribar a conclusiones.
- Pruebas estadísticas: Usadas para el análisis de los tiempos de ejecución obtenidos e inferir conclusiones estadísticamente válidas sobre la existencia de diferencias significativas entre las variantes del algoritmo, utilizando el paquete estadístico R Commander 2.0-3 (Rcmdr).

El **aporte práctico** de la investigación lo constituye un algoritmo de interpolación de Krigeado Ordinario, que ofrece menores tiempos en los procesos de estimación de reservas de minerales útiles, a partir del aprovechamiento eficiente de los recursos computacionales y el uso de métodos optimizados de búsqueda espacial. Como aporte social, la reducción de los tiempos asociados a la interpolación de Krigeado Ordinario soportará la toma de decisiones rápidas, durante la evaluación de la factibilidad de los proyectos, así como la planificación minera de estos en condiciones de rentabilidad económica.

Estructura del documento

La tesis se encuentra estructurada en resumen, introducción, tres capítulos como cuerpo fundamental de la tesis, conclusiones, recomendaciones, referencias bibliográficas y anexos.

En el *Capítulo I* "Fundamentos teóricos" se analiza la formulación matemática de la interpolación de Krigeado Ordinario; se realiza la modelación y análisis del algoritmo para identificar las oportunidades de optimización y paralelización en el proceso de interpolación de Krigeado Ordinario. Se analizan las ventajas y limitaciones de las soluciones previas de paralelización de algoritmos de interpolación de Krigeado; así como otros mecanismos utilizados para acelerar los tiempos de respuesta. Se identifican las tendencias actuales en la implementación de algoritmos paralelos de interpolación de los datos para la resolución de problemas de estimación.

En el *Capítulo II* "Materiales, métodos y resultados" se describen las técnicas de programación paralela en memoria compartida, así como las tecnologías o herramientas utilizadas. Se analizan técnicas de descomposición de los datos, asignación de tareas y balance de carga entre procesadores. Se proponen variantes del algoritmo de interpolación de Krigeado Ordinario, basadas en el uso de técnicas de programación paralela en memoria compartida, así como métodos optimizados de búsqueda espacial, y esbozadas a partir de los métodos de diseño existentes. Se realiza el análisis teórico de la complejidad temporal del algoritmo de

Krigeado Ordinario.

En el *Capítulo III* "Análisis de los resultados" se demuestra la validez de los resultados obtenidos mediante el algoritmo a partir de pruebas estadísticas para muestras relacionadas. Se comparan diferentes formas de particionar los datos y distribuir las tareas, identificando las formas apropiadas para la descomposición de los datos entre los procesadores. Se evalúa el rendimiento del algoritmo atendiendo a los tiempos de ejecución obtenidos para diferentes tamaños de entrada y número de procesadores; y también a partir de las métricas de análisis de algoritmos paralelos, entre ellas la ganancia de velocidad, la eficiencia y la escalabilidad. También se analizan cuestiones referentes al consumo de memoria.

Capítulo 1

Fundamentos teóricos

1.1 El Krigeado, sus variantes y aplicaciones

El Krigeado constituye un método de interpolación espacial muy utilizado en la construcción de superficies o cuerpos tridimensionales a partir de nubes irregulares de puntos; en la estimación de variables aleatorias en puntos no muestreados; así como en otras aplicaciones de la Geoestadística. Especialmente en el área de las Geociencias, es ampliamente utilizado en la estimación de recursos y reservas minerales útiles, considerando el nivel de precisión y confiabilidad que caracteriza sus resultados en estimaciones locales. Su objetivo es encontrar el Mejor Estimador Lineal Insesgado a partir de los datos disponibles (Chica, 1997), los cuales generalmente son insuficientes debido al costo de su obtención.

El Krigeado se basa en la intuición de que las variables espaciales de una determinada población se encuentran correlacionadas en el espacio, es decir, mientras más cercanos estén dos puntos sobre la superficie terrestre menor será la variación de los atributos medidos (Deraisme y De Fouquet, 1996). La formulación matemática del Krigeado aplica "una colección de técnicas generalizadas de regresión lineal para minimizar una varianza de estimación definida de un modelo a priori de covarianza" (Olea, 1991). Se apoya en variogramas como funciones estadísticas que expresan las características de variabilidad y correlación espacial del fenómeno que se estudia a partir de puntos muestreados.

El Krigeado tiene muchas variantes según los grados de estacionariedad de la función aleatoria que representa el fenómeno regionalizado: Krigeado Simple, Krigeado Ordinario, Krigeado Universal, Krigeado de Indicadores, Krigeado Gaussiano, entre otros (Apéndice B). Sin embargo, la variante más utilizada es el Krigeado Ordinario, a la cual está dirigida la presente investigación. El Krigeado Ordinario asume que la función aleatoria es estacionaria

de segundo orden[1] con media desconocida, lo cual indica la homogeneidad de las muestras en el área en la que se distribuye la variable, y además manifiesta que la correlación entre dos variables aleatorias depende únicamente de la distancia espacial que les separa y es independiente de su ubicación (Montero y Larraz, 2008).

1.2 Formulación matemática del Krigeado Ordinario

El problema del Krigeado Ordinario consiste en estimar el valor desconocido de un sitio expresado matemáticamente mediante la combinación lineal ponderada de los valores muestreados. A través del Krigeado Ordinario se puede estimar tanto el valor desconocido de un punto como el valor promedio de un bloque, conocidos respectivamente como Krigeado puntual o Krigeado de bloques, según el soporte de la medición de los datos (Díaz, 2002). La estimación se calcula mediante la expresión 1.1:

$$z_k = \lambda_1 z(x_1) + \lambda_2 z(x_2) + ... + \lambda_n z(x_n) \tag{1.1}$$

donde: z_k es el valor estimado, $z(x_i)$ son los valores de las ensayos, λ_i son los pesos de Krigeado, n es el número de observaciones disponibles.

El Krigeado atribuye un peso λ_i a la ley de cada muestra $z(x_i)$, donde los pesos altos corresponden a las muestras cercanas y los pesos débiles a las alejadas. La ponderación depende del modelo ajustado a los puntos medidos, la distancia a la ubicación de la predicción y a las relaciones espaciales entre los valores medidos alrededor de la ubicación de la predicción. Estos pesos se calculan considerando las características geométricas del problema, de manera que (Alfaro, 2007):

1. $\lambda_1 + \lambda_2 + ... + \lambda_n = 1$ se garantice la condición de universalidad (es decir la sumatoria de los pesos debe ser unitaria)
2. $\sigma_{E^2} = var[z^*(x_0) - z(x_0)]$ la varianza del error cometido sea mínima.

Estos elementos conducen a un problema de minimización con restricciones que se resuelve utilizando la técnica denominada multiplicadores de Lagrange. Este método involucra la incógnita auxiliar llamada parámetro de Lagrange (μ) y consiste en igualar a cero las derivadas parciales de la nueva función. Al realizar las *N+1* derivaciones se obtiene un sistema de *N+1* ecuaciones lineales con *N+1* incógnitas. Los valores de los pesos asociados a cada

[1]Estacionariedad de segundo orden (o estacionariedad débil): Se refiere a la homogeneidad de la media y la covarianza de la variable en estudio. En otras palabras, significa que el valor esperado de $z(x)$ debe ser constante para todos los puntos; y la función de covarianza entre 2 puntos x y $x + h$ depende solo del vector h y no del punto x.

uno de los puntos se calculan mediante la resolución de este sistema de ecuaciones (Ecuación 1.2) (Alfaro, 2007).

$$\begin{cases} \sum_{j=1}^{n} \lambda_j C(u_i - u_j) + \mu = C(u - u_i) \quad i = 1...n \\ \sum_{j=1}^{n} \lambda_j = 1 \end{cases} \tag{1.2}$$

El sistema de ecuaciones también puede ser expresado en forma matricial en función de la covarianza (Ecuación 1.3). Los términos del miembro izquierdo del sistema de ecuaciones se determinan mediante el cálculo de las covarianzas de cada par de ensayos. Por otra parte, el miembro derecho se determina mediante la covarianza entre el punto o bloque y cada uno de los ensayos. Se observa las propiedades simétricas de la matriz que conforma el miembro izquierdo del sistema de ecuaciones lineales. El aprovechamiento de esta propiedad permitirá reducir los tiempos de construcción de esta matriz.

$$\begin{pmatrix} C(u_1 - u_1) & \cdots & C(u_1 - u_n) & 1 \\ \vdots & \ddots & \vdots & \vdots \\ C(u_n - u_1) & \cdots & C(u_n - u_n) & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_n \\ \mu \end{pmatrix} = \begin{pmatrix} C(u_1 - u) \\ \vdots \\ C(u_n - u) \\ 1 \end{pmatrix} \tag{1.3}$$

En la interpolación de Krigeado Ordinario, el sistema de ecuaciones también puede ser expresado en función de la semivarianza[2] (Ecuación 1.4); considerando la relación existente entre ambos estadígrafos descrita por la Ecuación 1.5 (siendo h, la distancia entre dos muestras en una dirección determinada).

$$\begin{pmatrix} \gamma(u_1 - u_1) & \cdots & \gamma(u_1 - u_n) & 1 \\ \vdots & \ddots & \vdots & \vdots \\ \gamma(u_n - u_1) & \cdots & \gamma(u_n - u_n) & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_n \\ -\mu \end{pmatrix} = \begin{pmatrix} \gamma(u_1 - u) \\ \vdots \\ \gamma(u_n - u) \\ 1 \end{pmatrix} \tag{1.4}$$

$$C(h) = C(0) - \gamma(h) \tag{1.5}$$

La resolución del sistema de ecuaciones lineales consume la mayor parte del tiempo de ejecución, aproximadamente de un 96.0 % a un 99.8 %, según (Pesquer y Pons, 2011). La solución puede determinarse a partir de técnicas como el método Gauss-Jordan, conocido también como método de eliminación gaussiana, o puede resolverse mediante la Ecuación 1.6:

[2]Grado de autocorrelación que poseen los pares de puntos.

$$A = C^{-1} * D \tag{1.6}$$

donde: A es la matriz que contiene los factores de peso, C es la matriz que contiene las covarianzas entre las leyes de las muestras, C^{-1} es la matriz inversa de C y D es la matriz que contiene las covarianzas entre la ley del punto y las leyes de las muestras.

Una de las características más importantes del Krigeado Ordinario es que proporciona la varianza del error de estimación, la cual depende del modelo de variograma obtenido y de las localizaciones de los datos originales (Armstrong y Carignan, 1997). La varianza del error puede calcularse mediante la Ecuación 1.7:

$$\sigma^2_{KO} = \sigma^2 - \sum_{i=1}^{n} \lambda_i C(u_i - u) - \mu \tag{1.7}$$

1.2.1 Ventajas del Krigeado Ordinario

La interpolación de Krigeado Ordinario con respecto a otras técnicas de interpolación espacial (método inverso de la distancia, *splines*, regresión polinomial, entre otros), tiene las siguientes ventajas (Montero y Larraz, 2008):

- Además de tener en cuenta las características geométricas (como el número y configuración de las localizaciones observadas), considera la estructura de la correlación espacial que se deduce de la información disponible a través de las estructuras semivariográficas, dando lugar a estimaciones más fiables.
- Las ponderaciones no se calculan sobre la base de reglas arbitrarias, sino que depende del comportamiento de la función que representa la estructura de correlación espacial.
- Permite medir la precisión de las estimaciones a través de la varianza del error de estimación, pudiendo calcularse incluso antes de conocer los valores de las variables en los puntos a estimar.
- Constituye un interpolador exacto, lo que implica que en los puntos que forman parte de la muestra la estimación coincide con el valor observado, siendo por tanto, nula la varianza de estimación asociada.

1.3 Descripción del Krigeado Ordinario

El proceso de Krigeado Ordinario involucra 5 pasos fundamentales (Figura 1.1): búsqueda de los ensayos en la vecindad definida, con el fin de limitar el número de datos a utilizar en

la interpolación y evitar el trabajo con grandes sistemas de ecuaciones lineales; la creación de la matriz de puntos (matriz de Krigeado) y el vector independiente a partir del cálculo de la función semivariograma; el cálculo de los pesos asociados a los ensayos, considerada la operación más costosa computacionalmente; el próximo paso consiste en predecir el valor del punto o el valor promedio del bloque, según sea el caso; y por último calcular la varianza del error de estimación. Este conjunto de pasos se repite para cada uno de los puntos o bloques a estimar.

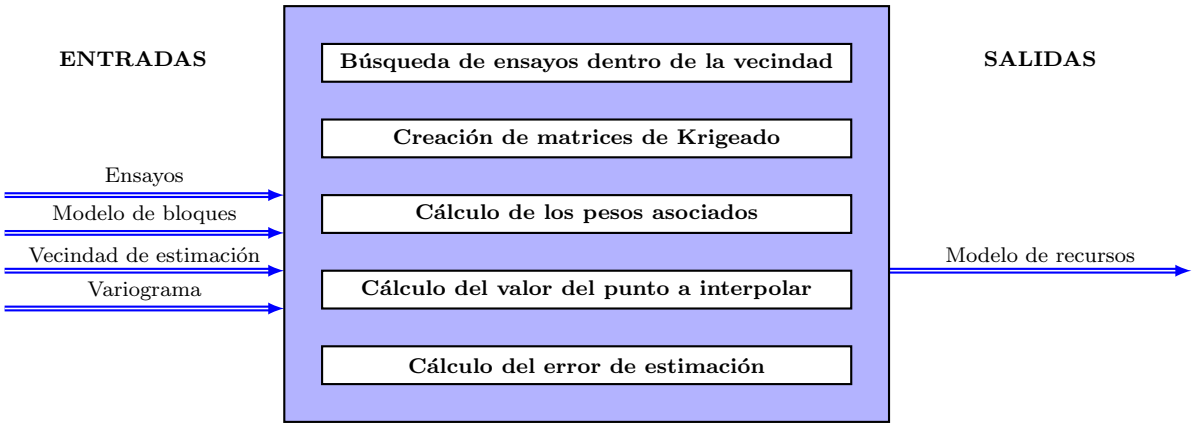


Figura 1.1: Proceso de Krigeado Ordinario: entradas y salidas.

El algoritmo de Krigeado Ordinario requiere como entradas:

- $A = \{a_1 \cdots a_n\}$ con $a_i = \{x_i, y_i, z_i, v_i\}$ ensayos o puntos medidos (representados por sus coordenadas espaciales y el valor del atributo medido).
- $M = \{b_1, b_2, \cdots, b_n\}$ modelo de bloques caracterizado por $p = \{(x_0, y_0, z_0), (nb_i * nb_j * nb_k), (lb_i * lb_j * lb_k), (db_i * db_j * db_k)\}$ (las coordenadas del origen, dimensiones del modelo, dimensiones de los bloques, nivel de discretización de los bloques); donde cada bloque contiene $b = \{(i, j, k), (x_c, y_c, z_c), (a_1, a_2, \cdots, a_n)\}$ (su localización espacial, las coordenadas del centroide y el conjunto de ensayos).
- $S = \{(R_x, R_y, R_z), (\alpha, \beta, \theta), (r_1, r_2, \cdots, r_n)\}$ vecindad de estimación delimitada por el elipsoide con radios R_x, R_y, R_z, los ángulos α, β, θ que indican la rotación del elipsoide en cada uno de los ejes, teniendo como restricciones: el número mínimo (r_1) y máximo de datos (r_2), así como el número máximo de datos por octante (r_3); representada por la Ecuación 1.8: donde x_0, y_0, z_0 constituyen las coordenadas del origen del centroide, y a, b, c los radios en cada uno de los ejes.

$$\frac{(x - x_0)^2}{a^2} + \frac{(y - y_0)^2}{b^2} + \frac{(z - z_0)^2}{c^2} = 1 \tag{1.8}$$

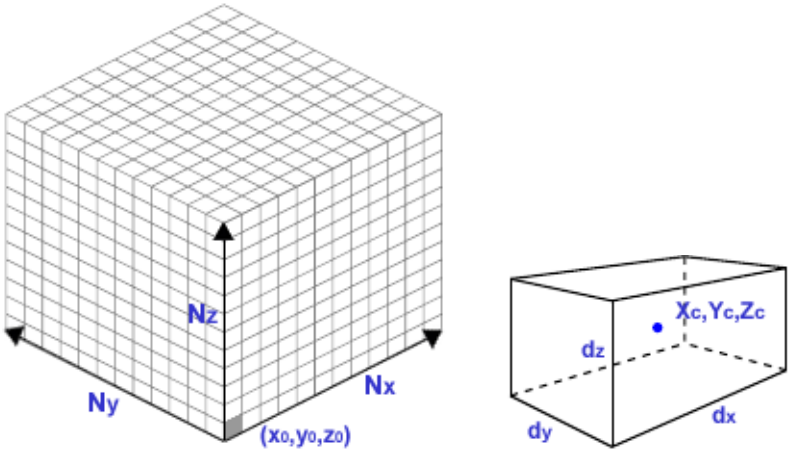
- $\gamma(h) = \{nst, c_0, (c_1\gamma_1(h) \cdots c_n\gamma_n(h))\}$ variograma que expresa las características de variabilidad y correlación espacial del fenómeno que se estudia a partir de puntos muestreados, siendo nst la cantidad de estructuras que conforman el variograma en cada una de las direcciones, c_0 el valor de pepita, c_i las mesetas, y $\gamma_i(h)$ las estructuras de variogramas.

El algoritmo genera como salidas:

- $M = \{b_1, b_2, \cdots, b_n\}$ modelo de recursos, es decir, los puntos o bloques estimados.

1.3.1 Modelo de bloques

Los modelos de bloques constituyen en la actualidad, la base fundamental para la estimación de las leyes minerales utilizando algoritmos de interpolación espacial asistidos por computadoras (Quesada, 2014). Esta técnica consiste en discretizar el yacimiento minero en bloques o celdas del mismo tamaño (Figura 1.2). Generalmente las dimensiones de los bloques coinciden con el tamaño de la unidad de selección minera que se utilizará durante la explotación del yacimiento, aunque las dimensiones óptimas dependen de otros factores como la variabilidad de las leyes, la continuidad geológica de la mineralización, el tamaño y espaciamiento de las muestras (Estévez, 2002).



(a) Modelo de bloques. (b) Bloque.

Figura 1.2: Modelo de bloques.

Los modelos de bloques se caracterizan por los siguientes parámetros: posición del modelo (coordenadas del centroide del bloque origen), extensión del modelo en las distintas direcciones XYZ, dimensiones de los bloques, y orientación del modelo (ángulo de inclinación y azimut) (Estévez, 2002). A cada bloque se le asignan atributos como la ubicación geográfi-

ca (posición del centroide), mediciones de leyes minerales, características litológicas, tipos de mineralización y otros parámetros geotécnicos (Lagos, 2011). El Krigeado es uno de los principales métodos utilizados en la estimación de las leyes minerales de los bloques.

1.3.2 Vecindad de estimación

En el proceso de Krigeado se recomienda establecer una vecindad de búsqueda para evitar el trabajo con grandes sistemas de ecuaciones lineales, que requieren altos tiempos de resolución. Los sistemas mineros reconocidos internacionalmente permiten la definición de una vecindad de búsqueda que limite el número de ensayos geoquímicos a utilizar en las estimaciones de Krigeado. El dominio de vecindad puede obtenerse con reducciones proporcionales en cada uno de los alcances o a través de cuadrantes u octantes (Cuador, 2002).

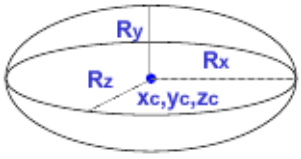


Figura 1.3: Elipsoide de búsqueda.

La vecindad del Krigeado se define como "el subconjunto del dominio que contiene el punto s_0 en el que se quiere estimar la función aleatoria y las localizaciones muestrales correspondientes a las observaciones que van a ser utilizadas en la estimación" (Montero y Larraz, 2008). La elección de la vecindad puede realizarse atendiendo a los siguientes criterios:

- *Criterio de vecindad única*: Se utilizan todas las observaciones disponibles en el dominio completo para estimar el valor de la función aleatoria en el punto deseado. Este criterio tiene como inconveniente el excesivo tiempo de computación necesario para la obtención de cada estimación cuando se dispone de un número elevado de observaciones.
- *Criterio de vecindad móvil (o local)*: Solo se utilizan las observaciones cercanas al punto donde se quiere realizar la estimación. Se debe definir el tamaño y la forma del vecindario, siempre centrado en el punto en el que se quiere realizar la estimación. La forma del vecindario se define teniendo en cuenta la anisotropía[3] de la función aleatoria, generalmente se utiliza un círculo o esfera para los casos isotrópicos y un elipse o elipsoide para los casos anisotrópicos. Se debe tener en cuenta la densidad del muestreo, es decir, que el número de datos por vecindario sea suficiente. Pueden considerarse otros aspectos como la distancia máxima, la distribución angular y la proximidad, para lo cual se suele

[3]El comportamiento es isotrópico si la variación de los datos es igual en todas las direcciones. El comportamiento es anisotrópico si la variación cambia con la dirección.

dividir la vecindad en sectores angulares (cuadrantes en 2D u octantes en 3D).

Al usar solo las observaciones de una vecindad cercana al punto a interpolar, se reduce significativamente las dimensiones de las matrices usadas para el cálculo de los pesos requeridos para predecir los valores en sitios no muestreados (Hessami et al., 2001). Sin embargo, la búsqueda de las observaciones cercanas en rejillas finas se considera computacionalmente exigente. Además, la existencia de discontinuidad en la predicción a lo largo de la periferia de las regiones locales afecta la interpolación (Vasan y Murtugudde, 2010).

1.3.3 Variograma

El variograma (o semivariograma), considerado como la herramienta fundamental de la geoestadística, permite a partir de la covarianza entre los puntos, representar la variabilidad de los mismos y su dependencia en función de la distancia y la dirección (Murillo et al., 2012). Su construcción se conoce como análisis estructural o variografía. Este proceso comienza con la representación gráfica del semivariograma empírico, calculado mediante la Ecuación 1.9 para todos los pares de ubicaciones separadas a una distancia h:

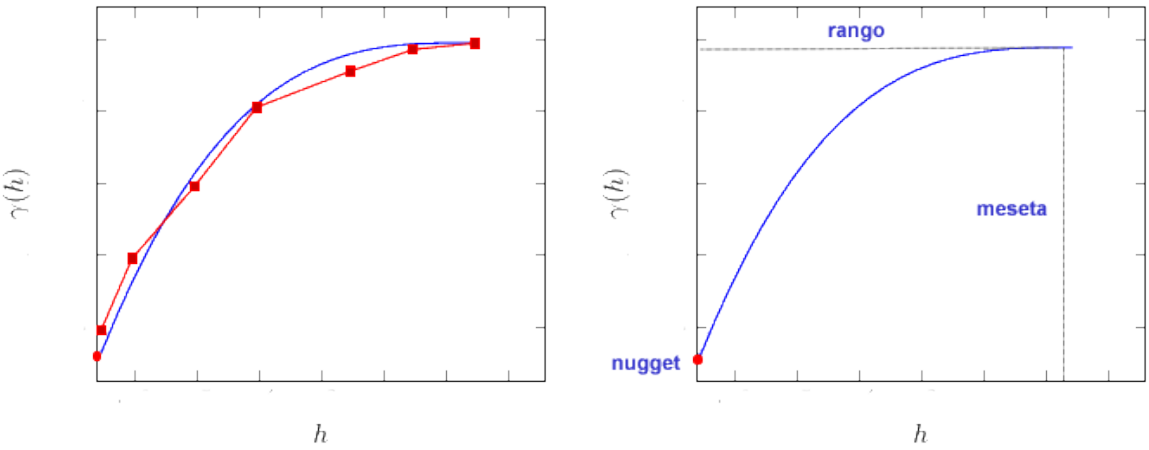
$$\gamma(h) = \frac{1}{2N(h)} \sum_{j=1}^{N(h)} [z(x_i + h) - z(x_i)]^2 \tag{1.9}$$

donde: $N(h)$ es el número de pares de muestras, h es la distancia o incremento, $z(x_i)$ son los valores de las variables muestreadas y x_i son las coordenadas espaciales de las variables.

El siguiente paso consiste en ajustar un modelo matemático a los puntos que conforman el semivariograma empírico (Figura 1.4a). Para realizar el ajuste se puede utilizar los siguientes modelos teóricos: circular, esférico, exponencial, gaussiano y lineal (Apéndice C). Estos modelos son descritos por: el rango, la meseta y el *nuggest*, los cuales se definen a continuación: (Figura 1.4b).

- *Rango*: Distancia a la que el modelo comienza a aplanarse; indica la distancia a partir de la cual las muestras son espacialmente independientes unas de otras.
- *Meseta*: Valor en el cual el modelo alcanza el rango (ESRI, 2012). Se considera la máxima semivarianza encontrada entre pares de puntos; se conoce como *sill* y debe coincidir con la varianza de la población (Gallardo y Maestre, 2008).
- *Nuggest*: Varianza no explicada por el modelo de variograma o discontinuidad de salto en el origen, y se calcula como la intercepción con el eje y (Gallardo y Maestre, 2008); este valor puede atribuirse a errores de medición o a fuentes espaciales de variación a

distancias que son menores que el intervalo de muestreo (ESRI, 2012).



(a) Ajuste del variograma empírico a un modelo teórico de variograma.

(b) Partes del variograma.

Figura 1.4: Variogramas.

1.3.4 Pasos del Krigeado Ordinario

A continuación se explica detalladamente el procedimiento del Krigeado Ordinario:

1. Localizar los n ensayos que van a contribuir al cálculo de la ley según la vecindad y los parámetros de búsqueda definidos.
2. Calcular la varianza de la ley del bloque σ_v: se determina promediando el valor de la función variograma $\gamma(h)$ para todos los pares de puntos interiores al bloque (solo en el Krigeado de bloques).
   - Para cada par de puntos interiores al bloque:
     . Calcular la distancia h entre los puntos
     . Calcular el valor de la función variograma $\gamma(h)$
     . Calcular el valor promedio de la función variograma $\bar{\gamma}(h)$
   - Calcular la covarianza entre los puntos $\sigma(h)$
3. Calcular las covarianzas entre las leyes de los ensayos σ_{x_i,x_j}: las cuales constituyen los términos del miembro izquierdo del sistema de ecuaciones.
   - Para cada par de ensayos
     . Calcular la distancia h entre los ensayos
     . Calcular el valor de la función variograma $\gamma(h)$
     . Calcular la covarianza entre los ensayos $\sigma(h)$

4. Calcular las covarianzas entre la ley del bloque y las leyes de los ensayos σ_{v,x_i} (Krigeado de bloques) o las covarianzas entre la ley del punto y las leyes de los ensayos σ_{x,x_i} (Krigeado puntual). Constituyen los términos del miembro derecho del sistema de ecuaciones.
    - Para cada ensayo
        - Para cada punto interior al bloque
            . Calcular la distancia h entre los puntos
            . Calcular el valor de la función variograma $\gamma(h)$
        - Calcular el valor promedio de la función variograma $\bar{\gamma}(h)$
        - Calcular la covarianza bloque-ensayos σ_{v,x_i} o covarianza punto-ensayos σ_{x,x_i}.
5. Determinar los factores de pesos λ_i mediante la resolución del sistema de ecuaciones de Krigeado.
6. Calcular la estimación $z(x)$ si se cumple la cantidad mínima de ensayos.
7. Calcular la varianza del error de estimación σ_e^2.

Los Diagramas 1.5 y 1.6 modelan de manera simplificada el proceso de Krigeado Ordinario descrito anteriormente.

1.4 Análisis de software de geoestadística

La cantidad de cálculos que implica el uso de los métodos geoestadísticos hace imprescindible la utilización de aplicaciones informáticas para el proceso de estimación. En el ámbito internacional existen diversos programas informáticos profesionales que aplican los métodos geoestadísticos para la estimación de las reservas, entre estos se puede mencionar: DATAMINE, GEMCOM, VULCAN, MICROMINE, GEOPACK, SURPAC, GEOSTAT, MICROLYNX, MINEMAP (Cuador, 2002). En el país se han introducido pocos sistemas mineros, debido fundamentalmente a los altos costos por conceptos de adquisición, actualización y capacitación; y también por causa del bloqueo económico impuesto por el gobierno norteamericano.

Numerosas herramientas se encuentran disponibles para la realización de análisis geoestadísticos sobre los datos, tanto alternativas libres como comerciales. Una revisión general de los software de análisis geoestadísticos disponibles en la actualidad, se encuentra publicada por AI-GEOSTATS en *52 North exploring horizonts* (AI-GEOSTATS, 2011). Las aplicaciones comerciales constituyen soluciones integrales de un mismo proveedeor desarrolladas para la actividad minera y cuentan con soporte y actualización constante; en cambio las

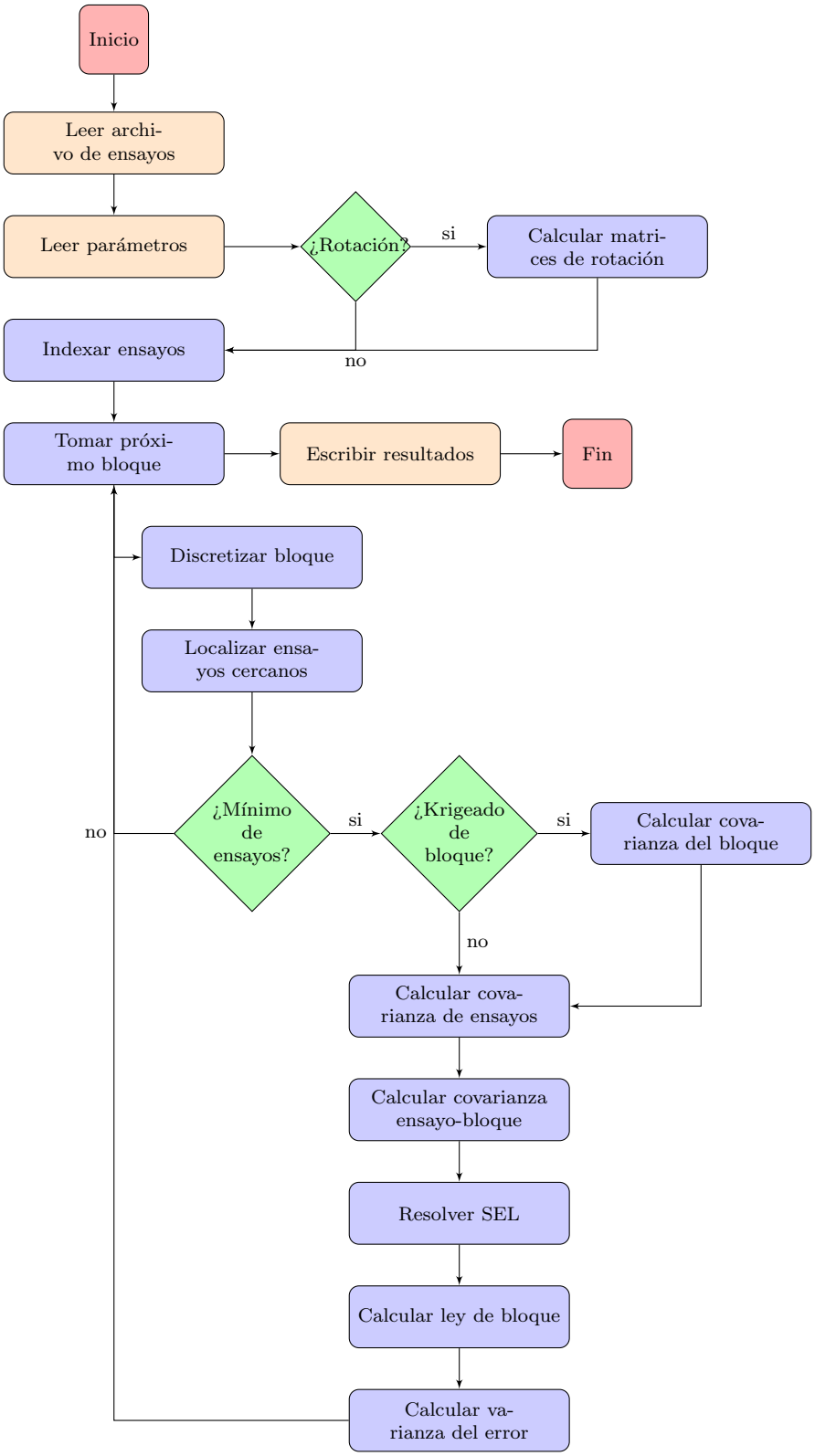


Figura 1.5: Diagrama de flujo del Krigeado Ordinario.

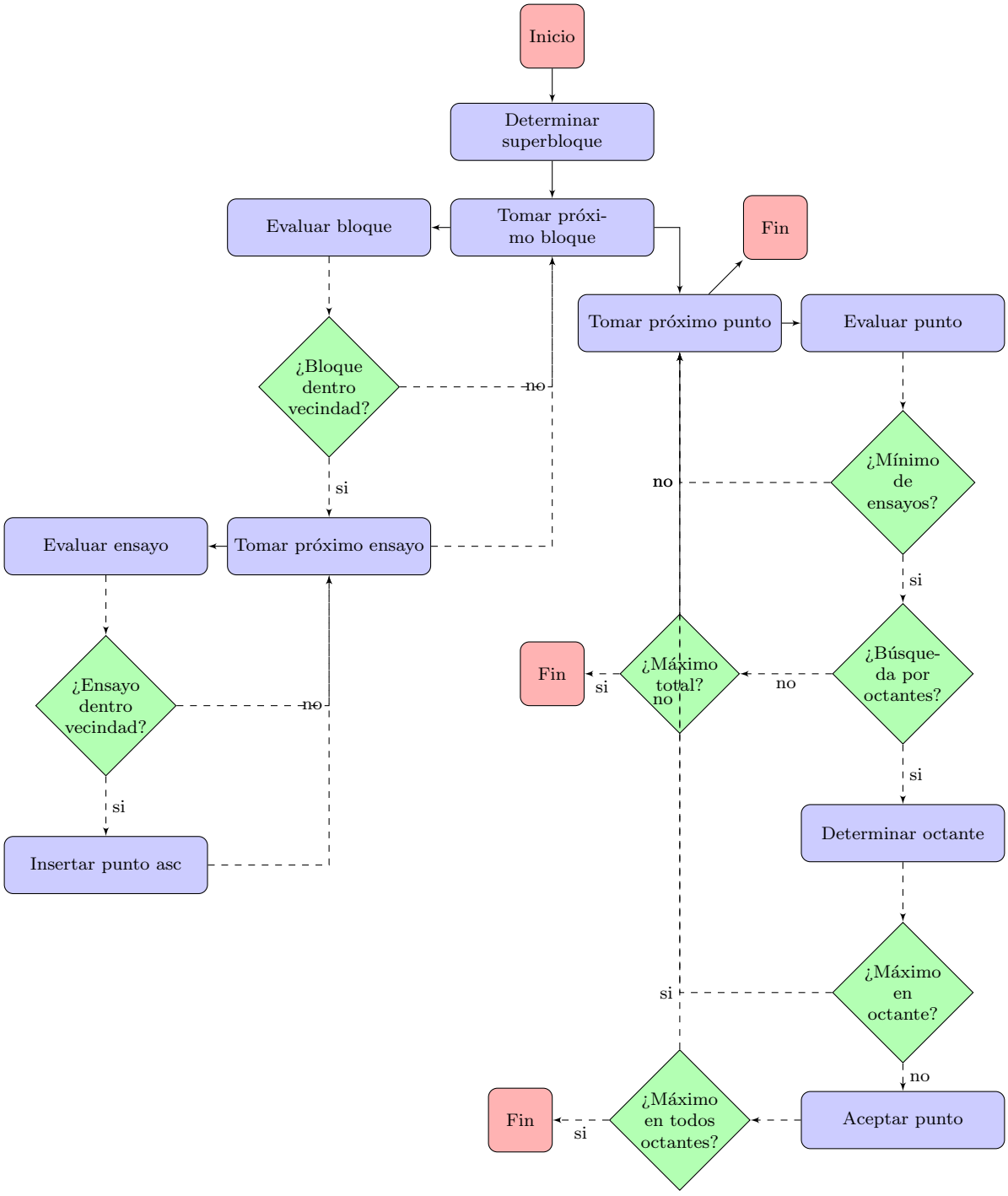


Figura 1.6: Diagrama de flujo de la búsqueda espacial de ensayos.

soluciones libres constituyen aplicaciones de uso parcial en la minería cuya actualización y soporte es escaso. En su mayoría estas soluciones operan sobre plataformas Windows con datos bidimensionales.

Entre las alternativas *open source* se encuentra GSLib (*Geoestatistical Software Library*), colección de programas geoestadísticos desarrollados en lenguaje Fortran por la Universidad de Stanford (Deutsch y Journel, 1997). S-Gems (*Stanford Geoestatistical Modeling Software*) desarrollada en C++ también por esta universidad, constituye el sucesor natural de GSLib, para la modelación geoestadística en 3 dimensiones. Este software se caracteriza por contar con una interfaz fácil de usar, una visualización interactiva en 3D y una amplia selección de algoritmos (Remy et al., 2009). Actualmente, S-Gems auxilia a soluciones mineras libres como RecMin en la realización de análisis exploratorios y variográficos de datos, así como estimaciones de recursos.

Isatis es una de las alternativas comerciales reconocida durante más de 20 años por su nivel de confiabilidad en las estimaciones de recursos, incluyendo la cuantificación de los riesgos asociados. Se caracteriza por ofrecer herramientas que cubren todos los aspectos de la geoestadística minera, entre las que se encuentran herramientas para: análisis exploratorio de datos, optimización de muestreo, análisis de la vecindad de búsqueda de Krigeado, técnicas multivariadas para estimación de recursos (Krigeado, co-Krigeado, Krigeado no lineal), entre otras. Las últimas versiones toman ventaja de los procesadores multinúcleos y soportan tarjetas gráficas como NVIDIA (Geovariances, 2016).

1.5 Análisis de tecnologías de computación paralela

Diversas estrategias han sido aplicadas para reducir los altos tiempos de ejecución de los métodos de interpolación de Krigeado. En la Web se identifican soluciones simples basadas en acelerar la búsqueda de las observaciones cercanas (Hessami et al., 2001), así como formas optimizadas de resolución de los sistemas de ecuaciones lineales (Allombert et al., 2014); y más recientemente, el uso de redes neuronales y la aplicación de técnicas de programación paralela, ya sea en entornos de memoria compartida o en arquitecturas distribuidas.

Las técnicas de programación paralela y distribuida han demostrado ser una alternativa viable para la solución rápida de este tipo de problemas computacionales. Dentro de los enfoques paralelos se identifican dos categorías: el uso de lenguajes de programación paralela o llamadas a las bibliotecas paralelas en máquinas fuertemente acopladas, y la descomposición del dominio de la nube de puntos a interpolar (Lu, 2004). El auge que ha tomado la programación paralela en las últimas décadas, ha posibilitado contar con diversidad de tecnologías

y dispositivos para una distribución de tareas con un mejor aprovechamiento de los recursos disponibles.

Muchas de las soluciones propuestas para la aceleración de los cálculos en la interpolación de Krigeado, aprovechan la escalabilidad que ofrecen las técnicas de computación paralela en entornos distribuidos. Dígase: supercomputadoras o clústeres (Kerry y Hawick, 1998), distribución de cálculos usando MPI (*Message Passing Interface*) en entornos de alto rendimiento (HPC, *High Performance Computing*) (Lu, 2004) (Pesquer y Pons, 2011) y cálculos de geoprocesamiento sobre la nube aplicando técnicas de *Map/Reduce* (Osorio, 2011).

Desafortunadamente, en el país son pocas las instituciones que disponen de entornos de alto rendimiento; y en muchas empresas el número de estaciones de trabajo es limitado. Por otro lado, el desarrollo de la computación de altas prestaciones sobre la nube es prácticamente nulo. Considerando estos elementos se descarta este tipo de soluciones en la presente investigación.

Publicaciones recientes señalan la utilización de GPU (*Graphic Processing Unit*) como una alternativa eficiente para la interpolación de Krigeado en sus distintas variantes, describiéndose muy buenos resultados de aceleración en (Vasan y Murtugudde, 2010), (Cheng, 2013), (Allombert et al., 2014) y (Gutiérrez de Ravé et al., 2014). La optimización de operaciones matriciales y las posibilidades de procesamiento paralelo masivo que ofrecen estos dispositivos, respaldan estos resultados. La computación general sobre unidades de procesamiento gráfico se identifica como una tendencia para abordar problemas de interpolación.

Sin embargo, la cantidad limitada de memoria disponible en estos aceleradores constituye un cuello de botella (Allombert et al., 2014). La utilización de este tipo de dispositivo de hardware obliga a programar para fabricantes específicos, añadiendo mayor complejidad en la programación. Estos dispositivos se consideran costosos en el ámbito nacional, y específicamente en el centro en el cual se desarrolla la investigación, no se dispone de ninguno. Estos elementos conducen a descartar la utilización de GPU en la investigación.

El paralelismo de datos en esquemas de memoria compartida constituye otra estrategia para abordar los problemas de interpolación. En (Cheng et al., 2010) se propone una versión paralela del Krigeado universal basado en OpenMP mediante un enfoque incremental. Este enfoque incremental se logra a través de los bucles iterativos que caracterizan el modelo de ejecución *fork-join*. Las aplicaciones basadas en OpenMP son relativamente fáciles de implementar, simplemente colocando directivas alrededor de los bucles, manteniéndose la mayor parte del código secuencial.

Las tendencias actuales en el desarrollo de hardware están encaminadas a la incorporación de más núcleos en los procesadores, en lugar del incremento de la velocidad del reloj

de los procesadores. Hoy día es común encontrar sistemas multinúcleos en las empresas e instituciones que practican la geología y la minería, tanto en las grandes empresas como las pequeñas. Por tal motivo, el algoritmo propuesto estará dirigido al aprovechamiento eficiente de las capacidades de procesamiento de los sistemas multinúcleos, adaptándose a las características del hardware subyacente. Esta adaptación permite obtener mejores rendimientos mientras mayores sean las capacidades de procesamiento paralelo de los equipos (Campbell y Miller, 2011).

1.6 Conclusiones parciales

- La complejidad de los cálculos matemáticos utilizados en la interpolación de Krigeado Ordinario, fundamentalmente la resolución de grandes sistemas de ecuaciones, repercute en altos tiempos de ejecución que retardan los procesos de estimación de recursos y reservas minerales.
- La confiabilidad de las estimaciones de Krigeado con respecto a otras técnicas de interpolación espacial, está dada por el aprovechamiento de la información disponible a través de estructuras semivariográficas, así como la posibilidad de medir la precisión de las estimaciones a través de la varianza del error de estimación.
- El estudio general de los software de análisis geoestadísticos evidenció que los software comerciales constituyen soluciones integrales desarrolladas para la actividad minera, con soporte y actualización constante, por lo que se encuentran en ventaja con respecto a las alternativas libres.
- La capacidad de adaptación al número de procesadores disponibles, la posibilidadades de los bucles iterativos paralelos y la facilidad de implementación que caracterizan los esquemas de memoria compartida, los convierten en una alternativa factible para abordar los problemas de interpolación de grandes conjuntos de datos.

Capítulo 2

Materiales, métodos y resultados

2.1 Computación paralela

El paralelismo se ha empleado durante muchos años, principalmente en la computación de altas prestaciones; pero en los últimos años ha experimentado un notable crecimiento como consecuencia de las limitaciones físicas del hardware. Las aplicaciones de la computación paralela se han extendido a diversas áreas de la ciencia, para abordar problemas que requieren gran capacidad de cómputo, por el volumen de datos que manejan o por la necesidad de respuesta en tiempo real. Numerosas aplicaciones científicas involucran procedimientos matemáticos que implican la resolución de sistemas de ecuaciones de gran tamaño, siendo la interpolación de datos espaciales uno de los casos.

La computación paralela consiste en "el uso de múltiples núcleos o procesadores simultáneamente para mejorar la velocidad de las aplicaciones" (Campbell y Miller, 2011). La paralelización se basa en la división en pequeñas partes independientes entre sí, de manera que estas puedan ejecutarse simultáneamente. En otras palabras, la computación paralela puede definirse como el uso de múltiples procesadores simultáneamente para solucionar problemas que requieren una alta capacidad de procesamiento, el objetivo es acelerar los procesos de cómputo.

Una computadora paralela consiste en un conjunto de procesadores que son capaces de trabajar de forma cooperativa para resolver un problema computacional (Foster, 1995). De acuerdo al nivel de paralelismo soportado, las computadoras paralelas se pueden clasificar en: sistemas multinúcleos, multiprocesamiento simétrico, clústeres, procesamiento paralelo masivo, computación distribuida y cómputo de propósito general en unidades de procesamiento gráfico.

2.1.1 Programación paralela en memoria compartida

En la computación paralela existen dos paradigmas fundamentales atendiendo al intercambio de datos entre las tareas paralelas: el paralelismo en memoria compartida (espacio compartido de direcciones) y el paralelismo en memoria distribuida (intercambio de mensajes), aunque también puede darse el caso de arquitecturas híbridas que combinan ambos paradigmas en busca de una mayor aceleración. Esta investigación se interesa en el paralelismo en memoria compartida.

En el paralelismo en memoria compartida la comunicación se basa en un recurso común; haciéndose necesario mecanismos que permitan sincronizar el acceso concurrente al recurso compartido (por ejemplo, semáforos y barreras). En este tipo de plataformas, el espacio compartido de direcciones puede ser local (exclusivo de un procesador) o global (común a todos los procesadores) (Grama et al., 2003). Sin embargo, el paralelismo en memoria distribuida se caracteriza por el paso de mensajes como mecanismo de comunicación entre los procesadores, cada uno con su propio espacio de direcciones.

La comunicación entre las tareas se realiza mediante operaciones de lectura-escritura sobre la memoria, de manera que los cambios efectuados en una localización de memoria son visibles para el resto de los procesadores. La comunicación entre los procesadores con la memoria se realiza a través de un bus o sistema de switches (llaves) de alta velocidad (Grama et al., 2003). La utilización de un espacio de memoria común ofrece mejor desempeño en comparación con los sistemas de memoria distribuida, referido al intercambio de información entre los procesos; además de que evita la duplicación de los datos.

Las plataformas de espacios compartidos de direcciones que soportan la programación SPMD se conocen como multiprocesadores. También se identifican diversas modelos de acceso a la memoria compartida: los modelos UMA (*Uniform Memory Access*) y NUMA (*Non-Uniform Memory Access*). En los modelos UMA la memoria física es compartida de manera uniforme por todos los procesadores, teniendo igual tiempo de acceso a todas las posiciones; mientras que en los modelos NUMA el tiempo de acceso a memoria depende de la posición a las cual se accede. COMA (*Cache-Only-Memory-Access*) se considera un caso especial de los modelos NUMA, en el cual las memorias compartidas principales se utilizan en forma de caché en vez de memoria principal real (Grama et al., 2003).

En los modelos de memoria compartida pueden ocurrir problemas de secciones críticas ante los cuales es necesario proteger el acceso a los recursos compartidos. Las soluciones a estos problemas deben garantizar las propiedades de exclusión mutua (a lo sumo un proceso está en la sección crítica), ausencia de deadlock (si dos o más procesos tratan de entrar a su sección crítica, al menos uno tendrá éxito), ausencia de demora innecesaria, y eventual

entrada.

2.1.2 Clasificación de sistemas paralelos según el mecanismo de control

La clasificación de sistemas paralelos propuesta por (Flynn, 1972) se basa en el número de instrucciones concurrentes, el flujo de datos y la manera en que son ejecutadas las instrucciones sobre los datos. Define las siguientes categorías:

- *Single Instruction stream, Single Data stream* (SISD): Un flujo único de instrucciones se ejecuta sobre un único conjunto de datos almacenados en una única memoria. Ejemplos: máquinas con uni-procesador o monoprocesador tradicionales
- *Single Instruction stream, Multiple Data stream* (SIMD): Un flujo único de instrucciones se ejecuta sobre diferentes conjuntos de datos. Ejemplos: porcesador vectorial o arreglo de procesadores.
- *Multiple Instruction stream, Single Data stream* (MISD): Diferentes flujos de instrucciones se ejecutan sobre el mismo conjunto de datos. Se usa en situaciones de paralelismo redundante, como por ejemplo en navegación aérea, donde se necesitan varios sistemas de respaldo en caso de que uno falle.
- *Multiple Instruction stream, Multiple Data stream* (MIMD): Diferentes flujos de instrucciones se ejecutan sobre diferentes conjuntos de datos. (En la práctica este tipo de organización nunca ha sido utilizada)

  La categoría MIMD ha sido extendida por otros autores:
- *Single program, multiple data* (SPMD): Cada procesador ejecuta una copia exacta del mismo programa, pero opera sobre datos diferentes.
- *Múltiples programas, múltiples datos* (MPMD): Múltiples procesadores autónomos que trabajan simultáneamente sobre al menos dos programas independientes.

De acuerdo a la clasificación de sistemas paralelos descrita por Michael J. Flynn, la interpolación de Krigeado Ordinario se ubica en la categoría *Single Instruction stream, Multiple Data stream* (SIMD, "una instrucción, múltiples datos"), la cual consisten en la ejecución de un único flujo de instrucciones sobre diferentes conjuntos de datos. A través de esta técnica se puede implementar paralelismo a nivel de datos. Se caracteriza por una única unidad de control que distribuye las instrucciones a diferentes unidades de procesamiento pero con diferentes conjuntos de datos. Estas instrucciones son ejecutadas simultáneamente por todas las unidades de procesamiento.

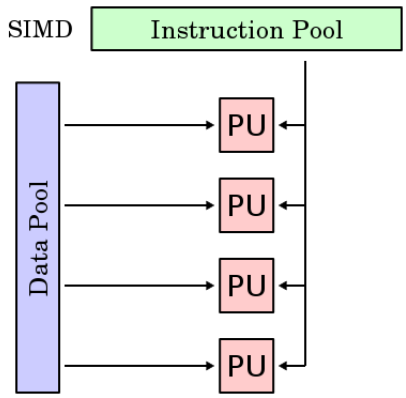


Figura 2.1: Single Instruction stream, Multiple Data stream (SIMD).

2.2 Diseño de algoritmos paralelos

En (Grama et al., 2003) se define un algoritmo paralelo como un conjunto de pasos para solucionar un problema usando múltiples procesadores. Diversos autores manifiestan la complejidad en la especificación de algoritmos paralelos, basándose en los problemas de concurrencia, entre ellas, las condiciones de competencia. Inicialmente se deben analizar los algoritmos de forma abstracta considerando solo los aspectos independientes de los equipos de cómputo, analizando en etapas posteriores los aspectos específicos. El diseño de algoritmos paralelos es una tarea difícil y creativa.

En (Grama et al., 2003) se plantean 5 principios a tener en cuenta en el diseño de algoritmos paralelos:

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

El diseño de algoritmos paralelos se basa en 4 etapas: partición (descomposición del cómputo en tareas), comunicación (coordinación en la ejecución de tareas), aglomeración (combinación de los resultados de las tareas) y mapeo (asignación de tareas a los procesadores) (Foster, 1995) (Naiouf, 2004).

- *Partición (Descomposición)*: El cómputo y los datos sobre los cuales se opera se descomponen en tareas. Se ignoran aspectos como el número de procesadores de la máquina a usar y se concentra la atención en explotar oportunidades de paralelismo. Se enfoca en definir un gran número de tareas pequeñas para obtener una descomposición de

granularidad fina. Existen dos formas de descomponer las tareas: descomposición del dominio (paralelismo de datos) o descomposición funcional (paralelismo de control o de tareas):

. 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.

. 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. Las formas de comunicación varían en dependencia de la arquitectura a utilizar. En arquitecturas de memoria distribuida cada tarea posee un identificador único y la comunicación entre estas se establece mediante el intercambio de mensajes; a diferencia de los ambientes de memoria compartida en los cuales no se produce envío de datos.
- *Agrupación (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. La idea es adaptar el algoritmo para que funcione eficientemente sobre determinados equipos. En esta etapa se consideran el incremento de granularidad[1] para reducir los costos de comunicación, la preservación de la flexibilidad con respecto a las decisiones de escalabilidad y la reducción de costos de ingeniería de software.
- *Asignación*: Cada tarea es asignada a un procesador tratando de maximizar la utilización de los procesadores y de reducir el costo de comunicación. Las políticas de asignación o mapeo de tareas tienen como objetivo central lograr el balance de carga entre los procesadores, el cual impacta directamente sobre el uso eficiente de los recursos, y por ende el rendimiento de las aplicaciones, en lo que se refiere a tiempo de ejecución. La asignación puede ser estática o dinámica.

  . La asignación estática se aplica cuando se conoce a priori la carga total de trabajo, y consiste en distribuir las tareas entre los procesadores al iniciar la ejecución del algoritmo. Si bien el balance estático es menos complejo que el balance dinámico, también es menos verśatil y escalable.

  . Al contrario, en la asignación dinámica la distribución de las tareas se realiza en

[1]La granularidad se determina por el número y el tamaño de las tareas en las que un problema es descompuesto. Se considera como granularidad fina la descomposición en un gran número de tareas de pequeño tamaño. Por el contrario, se considera granularidad gruesa la descomposición de un número pequeño de grandes tareas (Grama et al., 2003).

tiempo de ejecución a través de técnicas de balanceo de cargas. Estos métodos requieren alguna forma de mantener una visión global del sistema y algún mecanismo de negociación para la migración de procesos y/o datos. Si bien el balance dinámico tiene el potencial de mejorar el rendimiento global de la aplicación redistribuyendo la carga de trabajo entre los elementos de procesamiento, esta actividad se realiza a expensas de computación útil, produce sobrecarga de comunicación y requiere espacio en memoria para mantener la información.

2.3 Evaluación de algoritmos paralelos

La evaluación de algoritmos es fundamental para analizar la reducción de los tiempos de ejecución, y por ende, la calidad de las implementaciones en cuanto a rendimiento. Esta evaluación puede realizarse mediante análisis teóricos, mediante simulación o de forma experimental sin perder el contexto de la plataforma subyacente. El rendimiento de los algoritmos puede ser evaluado a partir de métricas que consideren el tamaño de entrada n y el número de procesadores p. Las métricas más comunes son: el tiempo de ejecución, la ganancia de velocidad, la eficiencia y la escalabilidad.

El tiempo de ejecución de un algoritmo puede variar significativamente de un ordenador a otro, dependiendo del hardware (memoria, reloj, procesador), del sistema operativo (multitarea, multiusuario), incluso del compilador y de la carga del sistema operativo. Por tal razón, contar las instrucciones requeridas por un algoritmo para procesar n elementos de entrada, se considera una medida del tiempo de ejecución, y se denomina **complejidad temporal** (Sil, 2009). Teóricamente, la complejidad temporal permite analizar la eficiencia de los algoritmos, así como establecer comparaciones entre algoritmos en términos de cómo estos escalan para grandes entradas de datos.

Se denomina **tiempo de ejecución** al tiempo que transcurre desde el comienzo de la ejecución del programa en el sistema paralelo, hasta que el último procesador culmine su ejecución (Grama et al., 2003). Denotado por la Ecuación 2.1:

$$T_P(n,p) = T_A(n,p) + T_C(n,p) + T_O(n,p) \tag{2.1}$$

Donde:

$T_A(n,p)$: Tiempo empleado por el algoritmo para realizar operaciones aritméticas (Cómputo)

$T_C(n,p)$: Tiempo empleado por el algoritmo para realizar comunicaciones entre los procesadores (Comunicación)

$T_O(n, p)$: Tiempo en que los procesadores se encuentran ociosos debido a desequilibrio de carga, sincronización u otros factores (Ocioso).

El rendimiento de los algoritmos en cuanto al tiempo de ejecución está limitado por factores como: la sobrecarga de sincronización y comunicación entre procesos, desequilibrio de carga, partes de código inherentemente secuencial, exceso de cómputo, máximo grado de concurrencia (Sanz, 2012) y el tamaño de la memoria del sistema.

2.3.1 Modelo *Work Span*

El modelo *Work Span* constituye una medida analítica para la evaluación de algoritmos paralelos, previamente a la implementación de estos. A diferencia de la complejidad temporal, las métricas *work* y *span* permiten analizar la escalabilidad de los algoritmos con respecto al número de procesadores.

Este modelo considera el algoritmo paralelo como una grafo acíclico dirigido, en el cual los nodos representan instrucciones y las aristas, dependencias entre las instrucciones. El número total de nodos en el grafo, se denomina *work* y el número de nodos a lo largo de la ruta más larga, se conoce como *span* (McCool et al., 2012). En caso de que los nodos tengan diferente costo, el *span* se determina mediante la suma de los costos en la ruta que posee la suma más alta .

Work se denota por T_1, corresponde al número total de instrucciones o tiempo de ejecución de un programa sobre un único procesador, mientras que *span* corresponde al tiempo de ejecución sobre una máquina ideal con un número infinito de procesadores, y es denotado por $T_{\text{ínf}}$. Al *span* también se conoce como la longitud o profundidad de la ruta crítica.

El modelo *Work Span* permite determinar el paralelismo, definido por $P = \frac{work}{span} = \frac{T_1}{T_{\text{ínf}}}$. El paralelismo denota la cantidad promedio de trabajo que puede ser realizado de forma paralela o simultánea. También se considera que el paralelismo indica la cantidad de procesadores que pueden ser utilizados eficientemente al ejecutar algoritmos paralelos. Representa la máxima ganancia de velocidad posible sobre cualquier número de procesadores (Ecuación 2.2) (McCool et al., 2012).

Este modelo considera que el trabajo total puede ser dividido en trabajo imperfectamente paralelizable $T_{\text{ínf}}$, y trabajo perfectamente paralelizable $T_1 - T_{\text{ínf}}$. Mientras el trabajo perfectamente paralelizable es escalable por el número de procesadores, el trabajo imperfectamente paralelizable debe ejecutarse estrictamente de forma secuencial, sin importar el número de procesadores disponibles. Por tanto se considera la Ecuación 2.3 como límite inferior de la ganancia de velocidad (McCool et al., 2012).

$$T_P(n,p) \leq \frac{T_1}{T_{\text{ínf}}} \tag{2.2}$$

$$T_P(n,p) \geq \frac{(T_1 - T_{\text{ínf}})}{p} + T_{\text{ínf}} \tag{2.3}$$

Considerando que cualquier cálculo ejecutable en paralelo sobre N procesadores, también puede ser ejecutado en $p \leq N$, si cada procesador ejecuta varias unidades de trabajo, la Ley de Brents establece la Ecuación 2.4:

$$\frac{T_1}{p} \leq T_P \leq \frac{T_1}{p} + T_{\text{ínf}} \tag{2.4}$$

2.3.2 Métricas de rendimiento de algoritmos

La **ganancia de velocidad** (*Speed-Up*) se define como la relación entre el tiempo de ejecución sobre un procesador secuencial y el tiempo de ejecución en múltiples procesadores. Mide la ganancia en rendimiento que se obtiene al resolver un problema de forma paralela con respecto a su implementación secuencial. Esta métrica permite medir el beneficio obtenido al emplear paralelismo. Está dada por la Ecuación 2.5:

$$S_P(n,p) = \frac{T(n,1)}{n,p} \tag{2.5}$$

La aceleración puede ser expresada de forma relativa: cuando el tiempo secuencial usado es el tiempo de ejecución paralelo sobre un único procesador; y absoluta: cuando el tiempo secuencial usado es el tiempo del mejor programa secuencial (Muller, 2011).

La Ley de Ambadahl permite expresar la máxima aceleración posible como una función de la fracción de la parte secuencial del algoritmo, que no puede ser paralelizada. Esto significa que para un problema dado existe un número óptimo de procesadores, que al ser aumentado conlleva a desperdiciar recursos computacionales (Gausellino et al., 2001).

Por otra parte, la Ley Gustafson Barsis rescata el paralelismo ante la Ley Ambadahl, al contrarestar las limitaciones impuestas por la parte secuencial de los programas con el aumento de la cantidad total de cómputo (Muller, 2011). Esta ley evidencia que problemas de mayor tamaño pueden ser resueltos en el mismo tiempo. A diferencia de la Ley de Ambadahl basada en problemas de tamaño fijo, la Ley de Gustafson Baris reconoce que los programadores están más interesados en maximizar el tamaño del problema al aumentar el número de procesadores disponibles en los equipos de cómputo.

Considerando $\frac{T(n,1)}{n,p}$ como la ganancia de velocidad sobre p procesadores, si:

- $\frac{T(n,1)}{n,p} = O(n)$ la ganancia de velocidad es lineal
- $\frac{T(n,1)}{n,p} = p$ la ganancia de velocidad es perfectamente lineal
- $\frac{T(n,1)}{n,p} \leq p$ la ganancia de velocidad es superlineal (óptimo).

Si la ganancia de velocidad es lineal, entonces se considera escalable.

La **eficiencia** mide el porcentaje del tiempo que los procesadores se mantienen útilmente empleados (Grama et al., 2003). Se considera una métrica de calidad y de costo del algoritmo (Gausellino et al., 2001). Se define por la Ecuación 2.6:

$$E(n,p) = \frac{S_P(n,p)}{p} \tag{2.6}$$

Se considera $E(n,p) = 1$ como máxima eficiencia con todos los procesadores trabajando, y $E(n,p) < p$ se considera como un bajo rendimiento (*slowdown*).

La **escalabilidad** es la capacidad de un determinado algoritmo de mantener sus prestaciones cuando aumenta el número de procesadores. Indica la capacidad del algoritmo de utilizar de forma efectiva un incremento en los recursos computacionales (Grama et al., 2003).

La escalabilidad puede ser evaluada de diferentes maneras, entre las que se encuentra el análisis de isoeficiencia en arquitecturas homogéneas. Este análisis plantea que un sistema es escalable si es posible mantener la eficiencia del sistema en un valor fijo al aumentar el número de procesadores (escalar la arquitectura) y el tamaño del problema (escalar el problema). La función de isoeficiencia indica cuánto tiene que aumentar el tamaño del problema para poder incluir más procesadores sin que la eficiencia del sistema se vea afectada (Sanz, 2012).

2.4 Indexación y búsqueda espacial por rangos

Con el objetivo de optimizar el algoritmo, evitando la realización de búsquedas innecesarias, se propone la búsqueda por rangos. La búsqueda por rangos, esencialmente consiste en buscar los objetos geométricos que contiene una determinada región del espacio de objetos geométricos (Olinda y Hernández, 2002). La eficiencia de las búsquedas por rangos se sustenta en la previa indexación de los objetos geométricos en una estructura de datos apropiada, en este caso considerando como estructura de datos el propio modelo de bloques.

El modelo de bloques se almacena en un arreglo multidimensional con almacenamiento lineal en el orden de las filas (*row major order*), esto quiere decir que los elementos consecutivos de las filas de la matriz son contiguos.

Para la indexación, por cada posición (i,j,k) del modelo de bloques se guarda una lista con los ensayos pertenecientes a la celda o bloque. Para determinar las celdas en las cuales se encuentran los ensayos, se transforman las coordenadas de los ensayos al espacio de coor-

denadas de los índices del modelo, teniendo en cuenta el origen (x_0, y_0, z_0) y las dimensiones (nb_i, nb_j, nb_k) de los bloques (Ecuación 2.7):

$$x_i = \frac{(x - x_0)}{lb_i}; y_j = \frac{(y - y_0)}{lb_j}; z_k = \frac{(z - z_0)}{lb_k}; \tag{2.7}$$

Luego se obtienen las localizaciones espaciales (i, j, k) de los ensayos redondeando por defecto las coordenadas transformadas: $i = floor\,(x_i)$, $j = floor\,(y_j)$ y $k = floor\,(z_k)$. Después se verifica que las localizaciones espaciales obtenidas estén dentro del rango de índices del modelo de bloques, cumpliéndose que $o \leq i \leq nb_i$, $o \leq j \leq nb_j$ y $o \leq k \leq nb_k$. Una vez indexados los ensayos, para buscar un ensayo en el modelo de bloques simplemente se calcula el índice de localización espacial del punto, siendo n_x la cantidad de columnas del modelo, n_y la cantidad de filas y xyz las coordenadas espaciales del punto en cuestión (Ecuación 2.8):

$$loc = zn_xn_y + yn_x + x \tag{2.8}$$

2.5 Descripción del algoritmo

2.5.1 Descomposición de los datos: patrón *parallel loops*

Aprovechando la independencia de los datos de entrada se aplica la descomposición del dominio para la división de los datos entre los procesadores. Esta técnica de descomposición consiste en determinar la partición apropiada de los datos, y luego trabajar en los cómputos asociados.

Se aplica el patrón *parallel loops* que consiste en aplicar una operación independiente a múltiples elementos de datos simultáneamente. Las iteraciones del bucle deben ser independientes, es decir, no pueden comunicarse unas con otras o escribir en locaciones de memoria compartida, que puedan ser accedidas por otras iteraciones (Campbell y Miller, 2011). Al utilizar este patrón, el grado de paralelismo no necesita ser expresado en el código ya que en tiempo de ejecución las iteraciones del bucle se ejecutarán simultáneamente según el número de procesadores disponibles. Este patrón es similar a los clásicos *for* o *for-each*, diferenciándose en que no se garantiza un orden particular en la ejecución de las iteraciones.

Se analiza el carácter independiente de las tareas y se identifica que en la tarea de indexación espacial, múltiples iteraciones pueden requerir acceder a un mismo bloque para ubicar los ensayos. En este caso, la aplicación de mecanismos de bloqueo para controlar el acceso a las variables compartidas, conduce a un cuello de botella, y no necesariamente los tiempos

disminuyen. Se genera una sección crítica, por lo que se concluye que para esta operación no es aplicable el patrón *parallel loops*. Por tanto, está porción del código se considera inherentemente secuencial, y limita la ganancia de velocidad. El resto de las operaciones si pueden beneficiarse del patrón.

Parallel loops también es conocido como paralelismo de datos, y forma parte del conjunto de patrones en torno al modelo *fork join* (Toub, 2010). *Fork join* constituye el principal modelo de ejecución paralela del estándar OpenMP. Este modelo plantea la división del hilo maestro en hilos esclavos que se ejecutan concurrentemente, distribuyéndose las tareas sobre estos hilos. Al final se sincronizan los hilos para asegurar que todo el procesamiento ha finalizado, se recuperan los resultados y continúa la ejecución el maestro. Estos hilos acceden a la misma memoria, aunque es posible gestionar estos accesos generando espacios de memoria privada (Chapman et al., 2008).

Las Figuras 2.2, 2.3 y 2.4 representan las diferentes estrategias de *fork-join* aplicadas en la investigación. En la primera estrategia los datos se distribuyen a los procesadores una sola vez, mientras que en la segunda estrategia, una vez realizada la búsqueda espacial, los datos se distribuyen nuevamente en aras de equilibrar la carga entre los procesadores. La última estrategia consiste en ordenar paralelamente los datos luego de realizar la búsqueda espacial, y también está diseñada para balancear la carga entre los procesadores.

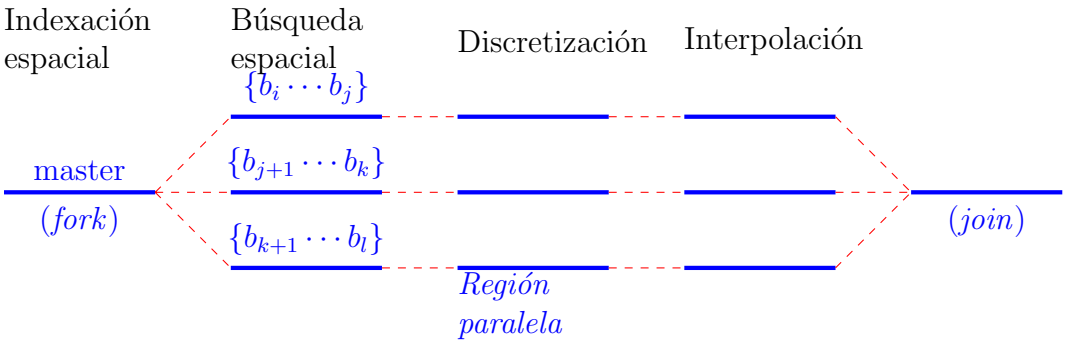


Figura 2.2: Estrategia de *fork-join* con una sola región paralela.

2.5.2 Asignación de las tareas: balance de carga y granularidad

A través de la claúsula $schedule(kind[, chunksize])$ se puede especificar la manera en que las iteraciones son distribuidas sobre los hilos de procesamiento. Esta claúsula ofrece 5 formas de asignación de los trozos de iteraciones (Chapman et al., 2008):

- *Estática*: Las iteraciones son divididas en trozos de tamaño *chunksize*. Los trozos son asignados a los hilos estáticamente de manera Round Robin. Cuando el *chunksize* no

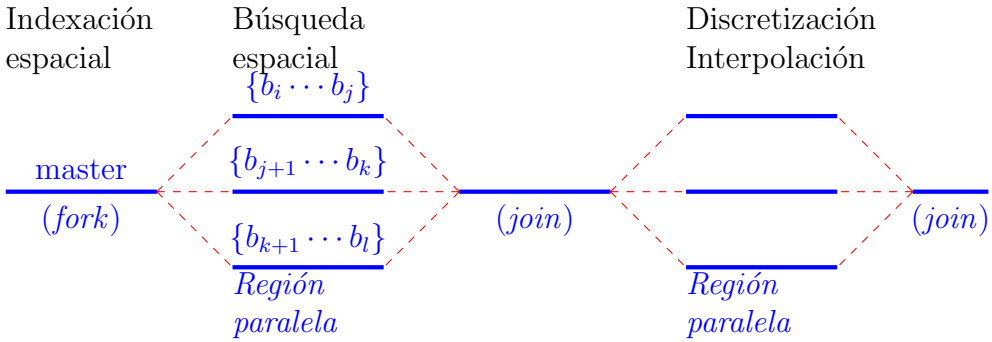


Figura 2.3: Estrategia de *fork-join* con 2 regiones paralelas.

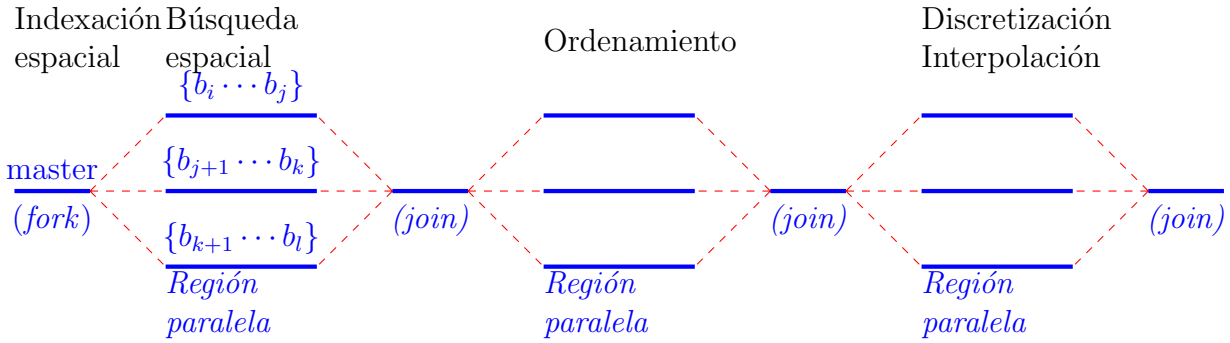


Figura 2.4: Estrategia de *fork-join* con 3 regiones paralelas.

es especificado el espacio de iteraciones es divido en partes de aproximdamente igual tamaño.

- *Dinámica*: Las iteraciones son divididas en trozos de tamaño *chunksize* y asignadas a los hilos en la medida en que estos lo soliciten. Por defecto, el *chunksize* toma valor 1.
- *Guiada*: Similar a la asignación dinámica, excepto que el *chunksize* decrementa cada vez que un trozo de iteraciones es asignado a un hilo. El tamaño de los trozos de iteraciones es proporcional al número de iteraciones sin asignar, dividido por el número de hilos. El parámetro *chunksize* define el mínimo tamaño de los bloques de iteraciones.
- *En tiempo de ejecución*: La programación o planificación se decide en tiempo de ejecución, mediante la variable global $OMP\_SCHEDULE$. No se permite especificar un *chunksize*.
- *Automática*: La programación o planificación de las iteraciones es delegada al compilador y/o sistema en tiempo de ejecución.

En los problemas de interpolación de Krigeado Ordinario se conoce a priori: la cantidad de bloques y ensayos, así como el nivel de discretización de los bloques, pero no se conoce el número de ensayos cercanos a utilizar en cada interpolación. El número de ensayos cercanos

que contribuirán a las estimaciones, depende de la vecindad de búsqueda, está limitado por el número mínimo de ensayos y el número máximo de ensayos permitidos; y no se conoce hasta que se realiza la búsqueda espacial de los mismos.

Cada bloque puede tener un valor distinto del número de ensayos cercanos, respetando siempre las restricciones de búsqueda; lo cual conduce a situaciones de desbalance de carga entre los procesadores. Al analizar el problema de balance de carga, se tiene en cuenta: la descomposición de los datos y el mapeo de tareas, así como cuestiones relativas al costo de las tareas, dependencias entre las tareas y localidad.

En cuanto al costo de las tareas, los procesadores realizarán las mismas tareas a elementos de datos diferentes, por lo que desde un punto de vista teórico, todos los procesadores realizarán tareas de igual complejidad computacional. En la práctica esto no sucede así, debido a que cada bloque puede estimarse a partir de cantidades diferentes de ensayos, parámetro que impacta cúbicamente los tiempos de ejecución.

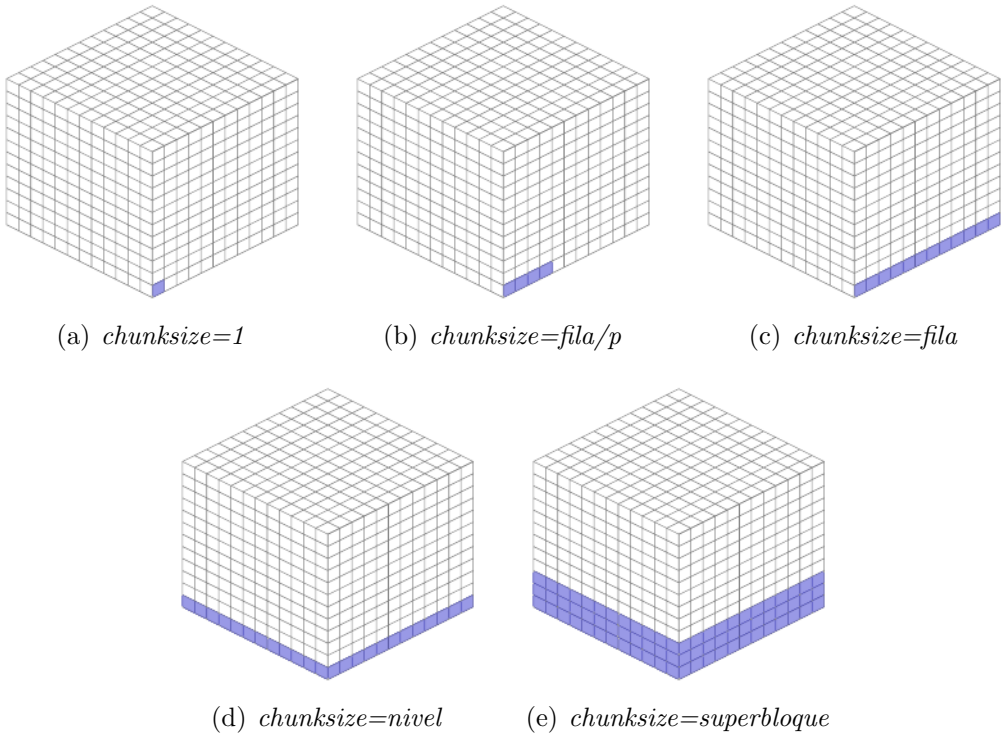
En cuanto a la dependencia entre tareas, la situación es favorable ya que los bloques pueden ser estimados en cualquier orden (*dependence free loops*). No se requiere comunicación entre las tareas que ejecutan distintos procesadores, por tanto estamos en presencia de lo que se conoce como *embarrassingly parallel*. Al no conocerse a priori la carga total de trabajo, una asignación estática probablemente no sea adecuada. Aunque para niveles bajos de discretización, y cantidades bajas de vecinos cercanos, las diferencias en las cargas de trabajo entre los procesadores, pudieran ser despreciables; pero esto depende de los datos entrados y no se puede asegurar.

Por otra parte, las asignaciones dinámica y guiada se consideran adecuadas cuando se tienen cargas de trabajo desiguales (como es el caso), ya que uno o más procesadores podrían estar ocupados por largo tiempo. Permiten a los procesadores con pequeñas cargas de trabajo, solicitar nuevos trozos de interaciones, en aras de que el trabajo se pueda distribuir de manera uniforme. Si embargo, este tipo de asignaciones tiene altos costos de sobrecarga (costos de sincronización por trozos), por lo que es recomendable seleccionar trozos de tamaños apropidados, que permitan disminuir la sobrecargas ocasionas cuando los procesadores solicitan más carga de trabajo, sin causar desequilibrios.

Una estrategia que combine las ventajas del *scheduling* estático y el dinámico debe ser capaz de generar una carga balanceada sin incurrir en un gran *overhead*. Esto es posible con técnicas avanzadas de *scheduling* paralelo. En el *scheduling* paralelo todos los procesadores cooperan para planificar el trabajo. Dado las características del Krigeado Ordinario, un ordenamiento de los bloques en función del número de vecinos, permitiría una distribución más equitativa de la carga entre procesadores, contribuyendo a un uso más eficiente de los

recursos computacionales.

La granularidad de la distribución de la carga de trabajo está definida por el *chunk*, el cual consiste en un subconjunto continuo del espacio de iteraciones. Se identificaron 5 formas de descomposición de los datos, que varian desde la simple planificación Round Robin, particiones aproximandamente equitativas y particiones del modelo de bloques, ya sea por fragmentos, filas, niveles o superbloques tridimensionales de forma horizontal. Se descartaron la descomposición de los datos en forma de columnas, niveles o superbloques verticales, considerando el orden fila principal en el que se encuentran almacenados los bloques.



(a) *chunksize=1* (b) *chunksize=fila/p* (c) *chunksize=fila*

(d) *chunksize=nivel* (e) *chunksize=superbloque*

Figura 2.5: Formas de partición de los datos.

Tabla 2.1: Variantes del algoritmo de Krigeado Ordinario.

Identificador	**Asignación**	***Chunksize***
1	Estática	1, fila/p, fila, nivel, superbloque horizontal
2	Dinámica	1, fila/p, fila, nivel
3	Guiada	1, fila/p, fila, nivel
4	Estática-dinámica	1, fila/p, fila, nivel
5	Estática-guiada	1, fila/p, fila, nivel
6	Estática-ordenamiento-dinámica	1, fila/p, fila, nivel
7	Estática-ordenamiento-guiada	1, fila/p, fila, nivel

2.5.3 Algoritmo para la interpolación de Krigeado Ordinario

El algoritmo de Krigeado Ordinario fue implementado usando C++11 como lenguaje de programación, OpenMP 4.8.2 como biblioteca de programación paralela en memoria compartida, y Atlas CLapack como biblioteca de algebra lineal optimizada para los cálculos matriciales. Esta biblioteca permitió simplificar el cálculo de operaciones matriciales como la solución de sistemas de ecuaciones lineales, así como la multiplicación de matrices.

A continuación se presenta el algoritmo en forma de pseudocódigo, según las 3 estrategias de *fork-join* definidas, incluyendo el pseudocódigo para interpolar un único bloque.

Algoritmo 1 Pseudocódigo del algoritmo de Krigeado Ordinario (modelo de bloques) según la estrategia I.

Require: A, M, S, $\gamma\,(h)$
Ensure: M
1: indexar (A, M)
2: $numbloques = M.cantidad$
3: **for parallel** $i = 0 : numbloques$ **do**
4: $b = M.at(i)$
5: discretizar (b)
6: $b.vecinos$ =buscar ($b.centroide$, A, M, S)
7: interpolar (b, A, M, S, $\gamma\,(h)$)
8: **end for**

Algoritmo 2 Pseudocódigo del algoritmo de Krigeado Ordinario (modelo de bloques) según la estrategia II.

Require: A, M, S, $\gamma\,(h)$
Ensure: M
1: indexar (A, M)
2: $numbloques = M.cantidad$
3: **for parallel** $i = 0 : numbloques$ **do**
4: $b = M.at(i)$
5: discretizar (b)
6: $b.vecinos$ =buscar ($b.centroide$, A, M, S)
7: **end for**
8: **for parallel** $i = 0 : numbloques$ **do**
9: interpolar (b, A, M, S, $\gamma\,(h)$)
10: **end for**

Algoritmo 3 Pseudocódigo del algoritmo de Krigeado Ordinario (modelo de bloques) según la estrategia III.

Require: A, M, S, $\gamma(h)$
Ensure: M

```
1: indexar (A, M)
2: numbloques = M.cantidad
3: for parallel i = 0 : numbloques do
4:    b = M.at(i)
5:    discretizar (b)
6:    b.vecinos =buscar (b.centroide, A, M, S)
7: end for
8: ordenar (M)
9: for parallel i = 0 : numbloques do
10:   interpolar (b, A, M, S, γ(h))
11: end for
```

Algoritmo 4 Pseudocódigo del algoritmo de Krigeado Ordinario (un bloque)

Require: b, A, M, S, $\gamma(h)$
Ensure: M

```
1: numvecinos = b.vecinos.cantidad
2: if numvecinos >= S.cantidadMinimaDeDatos then
3:    matLM = construirMatrizIzquierda (b, A, γ(h))
4:    matRM = construirMatrizDerecha (b, A, γ(h))
5:    matR = resolverSEL (LM, RM)
6:    valor = 0
7:    error = 0
8:    varianza = calcularVarianzaBloque (b, A, γ(h))
9:    μ = R[numvecinos]
10:   for i = 0 : numvecinos do
11:      valor+ = b.vecinos.at(i).valor
12:      error+ = R[i] * RM[i]
13:   end for
14:   b.valor = valor
15:   b.error = varianza − error − μ
16: else
17:   b.valor = NE {no estimado}
18:   b.error = NE {no estimado}
19: end if
```

2.6 Análisis teórico del algoritmo

Al analizar la complejidad temporal del algoritmo de Krigeado Ordinario se identificaron 4 operaciones fundamentales: indexación de los ensayos en el modelo de bloques, discretización del bloque a estimar, búsqueda espacial de los ensayos y finalmente, la interpolación. Denotando m como la cantidad de bloques, n como la cantidad de ensayos, v como la cantidad de vecinos que contribuirán a la estimación y d como el nivel de discretización de los bloques, a continuación se define la complejidad de cada una de las operaciones implicadas.

Tabla 2.2: Complejidad temporal.

Indexación:	$O(n)$
Discretización:	$O(d)$
Búsqueda:	$O(m_v * n_v)$
Interpolación:	$O(v^3 + d^2)$

La búsqueda espacial aplica un ordenamiento por inserción para la selección de los ensayos más cercanos en cada octante del elipsoide de búsqueda. El ordenamiento por inserción en el mejor de los casos tiene una complejidad $O(n)$, mientras que en el peor de los casos, tiene una complejidad $O(n^2)$ ya que todos los elementos deben ser reubicados.

Finalmente la complejidad temporal del algoritmo de Krigeado Ordinario está acotada por $O(m*(v^3+d^2))$, donde la cantidad de vecinos y la discretización tienen un fuerte impacto en los tiempos de ejecución. En el caso del Krigeado puntual la discretización de los bloques, conduce al mejor de lo casos ($O(1)$) ya que los bloques no se discretizan en puntos interiores; por tanto la complejidad del Krigeado Ordinario se simplifica a $\Omega(m * (v^3)$.

La ganancia de velocidad del algoritmo de Krigeado Ordinario está limitada por la porción de trabajo secuencial no paralelizable, que supone la indexación espacial. La operación de indexación genera regiones críticas que no pueden ser modificadas simultáneamente por varios procesadores, generando un cuello de botella. Por tanto, paralelizar la indexación puede conllevar un procesamiento adicional, que no mejora los tiempos de ejecución.

Teóricamente, el grado de paralelismo está dado por 2.9, y el nivel de flojedad está dado por 2.10. Cuando $n + m * (v^3 + d^2) < p * (n + (v^3 + d^2))$ no se podrá alcanzar una ganancia de velocidad perfectamente lineal.

$$\frac{O(n + m * (v^3 + d^2)}{O(n + (v^3 + d^2))} \tag{2.9}$$

$$\frac{O(n + m * (v^3 + d^2)}{p * O(n + (v^3 + d^2))}. \tag{2.10}$$

De acuerdo a la dificultad para resolver el algoritmo en términos de tiempo computacional, el algoritmo se considera de tipo P (complejidad polinomial) basado en modelo determinista (para una entrada produce un único resultado). Este tipo de algoritmo puede ser resuelto rápidamente.

2.7 Conclusiones parciales

- El carácter independiente de los datos utilizados en el proceso de Krigeado Ordinario favorece la utilización del paralelismo a nivel de datos como una alternativa eficiente para disminuir los tiempos de respuesta asociados.
- Las técnicas de programación paralela en memoria compartida facilitan la explotación del paralelismo de datos a través del uso de bucles iterativos para la distribución de tareas a los procesadores, propiciando un mejor aprovechamiento de las capacidades de cómputo.
- La subdivisión de los bloques en diferentes procesadores es muy útil cuando se trabaja con grandes cantidades de datos, pero si no se logra una distribución equitativa el aprovechamiento de los procesadores no será óptimo.
- Los métodos búsqueda espacial por rangos sustentados en la indexación espacial de los objetos geométricos contribuyen a disminuir los tiempos de respuesta en el proceso de Krigeado, al evitar búsquedas innecesarias.
- Si bien el ordenamiento de los datos facilita el particionado de los datos entre los procesadores de forma equilibrada, añade operaciones de procesamiento que repercuten en los tiempos de ejecución.

Capítulo 3

Análisis de los resultados

Los experimentos se ejecutaron en una estación de trabajo AsusTek H81M-K con procesador Intel (R) Core (TM) i7-4790 (compuesta por 4 núcleos físicos y 4 núcleos virtuales, para un total de 8 núcleos), con una frecuencia de 3.60 GHz, 8 GB de memoria instalada y sistema operativo Debian 7.3 (Wheezy) de 64 bits.

3.1 Validación de los resultados

Como caso de estudio para la validación de los resultados del algoritmo, se utilizaron datos de la región O48 pertenecientes al yacimiento laterítico de Punta Gorda en Moa (Figura 3.1). Se consideraron 1253 ensayos cuyas leyes satisfacen la condición: $Ni \geq 0{,}9$. Considerando el espacio cubierto por la mineralización, se generó un modelo de bloques con la siguiente geometría (obteniéndose un total de 19602 bloques):

- origen: 4733.33, 5533.34, 73.42
- dimensiones del modelo: 264, 264, 54 m
- dimensiones de los bloques: 8, 8, 3 m
- cantidad de bloques por eje: 33 x 33 x 18
- discretización de los bloques: 5 x 5 x 5.

Los radios de búsqueda se definieron considerando los alcances obtenidos en la dirección horizontal: 200, 200 y 3 m. Se definieron como restricciones de la vecindad: 2 ensayos como mínimo y 12 ensayos como máximo para realizar las estimaciones. A partir del análisis variográfico de la mineralización niquelífera, la ecuación de continuidad espacial definida es la Ecuación 3.1.

$$\gamma(h_x, h_y, h_z) = 0{,}04 + 0{,}13Esf(\frac{h_x}{36{,}663}, \frac{h_y}{36{,}663}, \frac{h_z}{9}) + 0{,}03Exp(\frac{h_x}{213{,}312}, \frac{h_y}{213{,}312}, \frac{h_z}{9}) \quad (3.1)$$

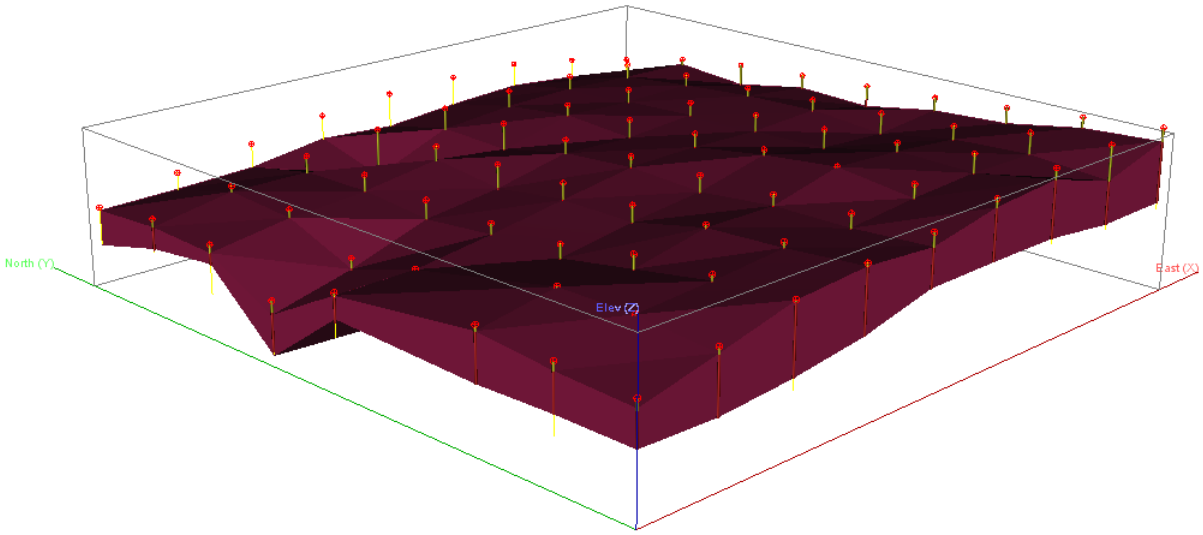


Figura 3.1: Pozos del bloque experimental O48 del yacimiento laterítico Punta Gorda.

Tabla 3.1: Comparación de los resultados de SGems y el algoritmo propuesto.

X	**Y**	**Z**	**Valor estimado**		**Error de estimación**	
			SGems	**Algoritmo**	**SGems**	**Algoritmo**
4733,33	5533,34	73,42	1,885	1,86696	0,366443	0,340951
4789,33	5533,34	73,42	1,885	1,945	0,366443	0,370046
4813,33	5541,34	73,42	1,885	1,945	0,366443	0,370046
4733,33	5533,34	79,42	1,34534	1,31106	0,287436	0,295597
4829,33	5533,34	79,42	1,295	1,28451	0,310055	0,286913
4861,33	5549,34	79,42	1,295	1,33132	0,310055	0,340951
4877,33	5565,34	79,42	1,3816	1,33132	0,300059	0,340951
4901,33	5597,34	79,42	1,3816	1,33132	0,300059	0,340951
4877,33	5661,34	79,42	1,31108	1,31131	0,282541	0,295549
4885,33	5661,34	79,42	1,34534	1,31131	0,287436	0,295552
4901,33	5685,34	79,42	1,34534	1,3113	0,287436	0,295555
4765,33	5693,34	79,42	1,39392	1,34319	0,275003	0,341164
4877,33	5693,34	79,42	1,39392	1,31133	0,275003	0,295544
4885,33	5701,34	79,42	1,31108	1,31132	0,282541	0,295548
4885,33	5741,34	79,42	1,34534	1,33132	0,287436	0,340951

La Tabla 3.1 muestra los resultados obtenidos para una muestra compuesta por 15 elementos. Los resultados obtenidos por el algoritmo propuesto fueron comparados con los resultados arrojados por el reconocido software SGems en su versión 2.1. El sistema SGems utiliza un método similar para la interpolación de Krigeado Ordinario; la comparación muestra un patrón de similitud en los resultados arrojados por ambas soluciones. Se demostró que las diferencias en los resultados están dadas porque ambas soluciones aplican diferentes estrategias de búsqueda de los ensayos cercanos que contribuirán a las estimaciones. Las leyes de los bloques o puntos no se calculan utilizando exactamente los mismos ensayos, los cuales influyen directamente en los valores estimados.

3.2 Comparación de las variantes del algoritmo

Se tienen 7 formas de asignación de las tareas (*estática, dinámica, guiada, estática-dinámica, estática-guiada, estática-ordenamiento-dinámica, estática-ordenamiento-guiada*), con 5 *chunksizes* (*1, fila/p, fila, nivel, partes iguales*); cada una con 10 mediciones de los tiempos de ejecución (variables cuantitativas). Estas mediciones se obtuvieron para el bloque experimental O48 usando de 2 a 100 ensayos como restricciones de la búsqueda. Los valores atípicos fueron identificados a partir del método de los cuartiles[1], y luego sustituidos por otras mediciones consideradas válidas, para no afectar las conclusiones derivadas de análisis estadísticos.

El tiempo promedio de ejecución para cada una de las variantes se muestra en las Tablas 3.2, 3.3 y 3.4. Un simple análisis de estas mediciones, permite observar que el balance estático en interpolaciones de Krigeado Ordinario se presenta como la peor forma de distribuir las tareas, ya que al no conocerse a priori la carga total de trabajo, esta se reparte de forma desigual entre los procesadores.

[1]Los cuartiles son los tres valores Q_1, Q_2 y Q_3 que dividen el conjunto de datos ordenados en cuatro partes porcentualmente iguales, determinan los valores correspondientes al 25 %, 50 % y al 75 % de los datos. La diferencia entre el tercer y primer cuartil se conoce como rango intercuartil. Los límites internos del conjunto de datos se calculan mediante $Q_1 - 1{,}5(Q_3 - Q_1)$ y $Q_3 + 1{,}5(Q_3 - Q_1)$, los valores fuera de estos límites se consideran levemente atípicos. Los límites extremos se calculan mediante $Q_1 - 3(Q_3 - Q_1)$ y $Q_3 + 3(Q_3 - Q_1)$, los valores fuera de estos límites se consideran valores atípicos extremos

Tabla 3.2: Mediciones de los tiempos de ejecución usando 2 procesadores.

Asignación	***Chunksizes***				
	1	**fila/p**	**fila**	**nivel**	**partes iguales**
Estática	47,6441	47,9209	47,7172	47,7068	56,4828
Dinámica	47,6093	47,5577	47,5604	47,8627	
Guiada	49,6065	49,5562	49,5427	49,5506	
Estática-Dinámica	47,974	48,0715	48,1403	48,9957	
Estática-Guiada	48,4261	48,5078	48,6703	48,397	
Estática-Ord-Dinámica	47,8455	47,8202	47,9582	49,8498	
Estática-Ord-Guiada	47,904	48,1297	47,9131	48,3549	

Tabla 3.3: Mediciones de los tiempos de ejecución usando 4 procesadores.

Asignación	***Chunksizes***				
	1	**fila/p**	**fila**	**nivel**	**partes iguales**
Estática	25,2089	25,3593	25,2259	26,5419	38,2231
Dinámica	25,0905	25,1708	25,1154	26,4563	
Guiada	26,3523	26,3818	26,2356	26,2572	
Estática-Dinámica	25,2059	25,232	25,4244	26,3235	
Estática-Guiada	25,2401	25,1817	25,1912	25,6957	
Estática-Ord-Dinámica	25,1233	25,2854	25,2224	27,9153	
Estática-Ord-Guiada	25,2726	25,2292	25,3076	29,8073	

Tabla 3.4: Mediciones de los tiempos de ejecución usando 8 procesadores.

Asignación	***Chunksizes***				
	1	**fila/p**	**fila**	**nivel**	**partes iguales**
Estática	20,9871	21,0851	21,1201	21,9463	25,0624
Dinámica	20,9773	20,9234	21,041	21,635	
Guiada	21,3441	21,271	21,2777	22,2661	
Estática-Dinámica	21,4364	21,4542	21,6535	22,1894	
Estática-Guiada	21,5294	21,4787	21,4999	22,4088	
Estática-Ord-Dinámica	21,4294	21,4053	21,4914	23,4303	
Estática-Ord-Guiada	21,4345	21,4909	21,4919	23,0194	

Considerando que los tiempos de ejecución constituyen variables cuantitativas de tipo ordinal, y considerando como grupos pareados las mediciones obtenidas para cada variante del algoritmo en iguales condiciones, se aplican las pruebas de Friedman[2] con el objetivo de demostrar la existencia de diferencias significativas entre las variantes analizadas en cuanto al tiempo de ejecución. Las variantes utilizadas en las pruebas de Friedman se restringuieron solo a aquellas combinaciones entre formas de asignación de tareas y *chunksizes* que arrojaron tiempos menores. Las Tablas 3.5, 3.6 y 3.7 muestran los 7 grupos pareados compuestos por 10 mediciones cada uno.

Tabla 3.5: Mediciones de los tiempos de ejecución de las variantes seleccionadas usando 2 procesadores.

Tiempos de ejecución						
Est	**Din**	**Gui**	**Est-Din**	**Est-Gui**	**Est-Ord-Din**	**Est-Ord-Gui**
47,7179	47,5468	49,4947	47,7113	48,3868	47,7433	47,8884
47,6246	47,5536	49,5545	47,9835	48,3967	47,7832	47,9555
47,5001	47,5983	49,4638	47,9143	48,4315	47,8895	47,8059
47,8718	47,5881	49,5641	48,447	48,3398	47,7896	47,9508
47,8059	47,5651	49,5429	48,3123	48,5318	47,7206	47,7563
47,5301	47,5402	49,5223	47,887	48,2646	47,9044	48,182
47,5725	47,5791	49,6512	47,7697	48,2609	47,9688	47,7096
47,4797	47,5826	49,5026	48,2277	48,2457	47,7881	47,8267
47,7457	47,5159	49,5628	47,7163	48,4724	47,9092	47,9935
47,5931	47,5073	49,569	47,7716	48,6403	47,7056	47,9715
Media						
47.60885	47.55935	47.60885	47.90065	48.39175	47.78885	47.91960

[2]Prueba no paramétrica para el análisis de varianza de grupos con muestras pareadas (no exige condiciones de normalidad ni igualdad de varianzas). La hipótesis nula es que los rangos sumados para cada columna (cada variable) sean iguales, es decir no existen diferencias significativas entre los grupos; y la hipótesis alternativa es que al menos uno es diferente, es decir si existen diferencias significativas entre los grupos. El estadístico de Friedman sigue una distribución de χ^2 con grados de libertad $k-1$, siendo k el número de variables relacionadas.

Tabla 3.6: Mediciones de los tiempos de ejecución de las variantes seleccionadas usando 4 procesadores.

Tiempos de ejecución						
Est	**Din**	**Gui**	**Est-Din**	**Est-Gui**	**Est-Ord-Din**	**Est-Ord-Gui**
25,2505	25,0742	26,2393	25,2817	25,1678	25,0145	25,1018
25,2103	25,0999	26,2367	25,1923	25,1896	25,1751	25,1889
25,1418	25,1475	26,1838	25,1696	25,2873	25,1187	25,2615
25,2596	25,0641	26,1866	25,2423	25,2801	25,0289	25,3044
25,1308	25,1139	26,2754	25,2381	25,2286	25,1046	25,2134
25,2501	25,042	26,279	25,1275	25,1488	25,1293	25,3389
25,1656	25,1081	26,2795	25,1786	25,0811	25,1926	25,3002
25,1968	25,0604	26,1997	25,2481	25,2446	25,2038	25,2506
25,2927	25,0939	26,1735	25,1884	25,0745	25,0642	25,1729
25,1916	25,1013	26,3033	25,1933	25,115	25,2017	25,1594
Media						
25.20355	25.09690	26.23800	25.19280	25.17870	25.12400	25.23200

Tabla 3.7: Mediciones de los tiempos de ejecución de las variantes seleccionadas usando 8 procesadores.

Tiempos de ejecución						
Est	**Din**	**Gui**	**Est-Din**	**Est-Gui**	**Est-Ord-Din**	**Est-Ord-Gui**
20,9377	20,9313	21,3273	21,4003	21,5226	21,4385	21,5736
20,9231	20,9199	21,2699	21,4743	21,4163	21,4078	21,4988
21,1337	20,9898	21,2358	21,4607	21,4886	21,3465	21,4745
20,9438	20,8848	21,2463	21,453	21,4203	21,4702	21,4581
20,9631	20,9892	21,3221	21,3489	21,4387	21,3771	21,5866
21,0582	20,9525	21,2652	21,4475	21,5489	21,5301	21,445
20,9144	21,0056	21,2498	21,4694	21,5579	21,4916	21,5071
20,9972	21,0667	21,2546	21,4621	21,4414	21,4476	21,4361
21,0247	21,0157	21,3078	21,4381	21,4736	21,3771	21,5033
20,9753	21,0179	21,299	21,4101	21,4796	21,4083	21,4267
Media						
20.96920	20.98950	21.26755	21.45025	21.47660	21.42340	21.48665

Los puntos críticos hallados para una distribución de χ^2 de Pearson con 9 grados de libertal para un nivel de confianza del 95 %, son 4.58E-006, 2.07E-003 y 3.91E-006 para 2, 4 y 8 procesadores respectivamente. Los valores obtenidos aplicando la fórmula de χ_r^2 de Friedman son 50.0571, 36.6429 y 50.4 para 2, 4 y 8 procesadores respectivamente. Como los valores

obtenidos aplicando la fórmula de χ_r^2 de Friedman son mayores, hay pruebas estadísticas suficientes para rechazar la hipótesis nula y concluir que existen diferencias significativas entre los tiempos de ejecución de las 7 variantes analizadas. Se deduce que al menos una variante arroja mejores tiempos que las demás.

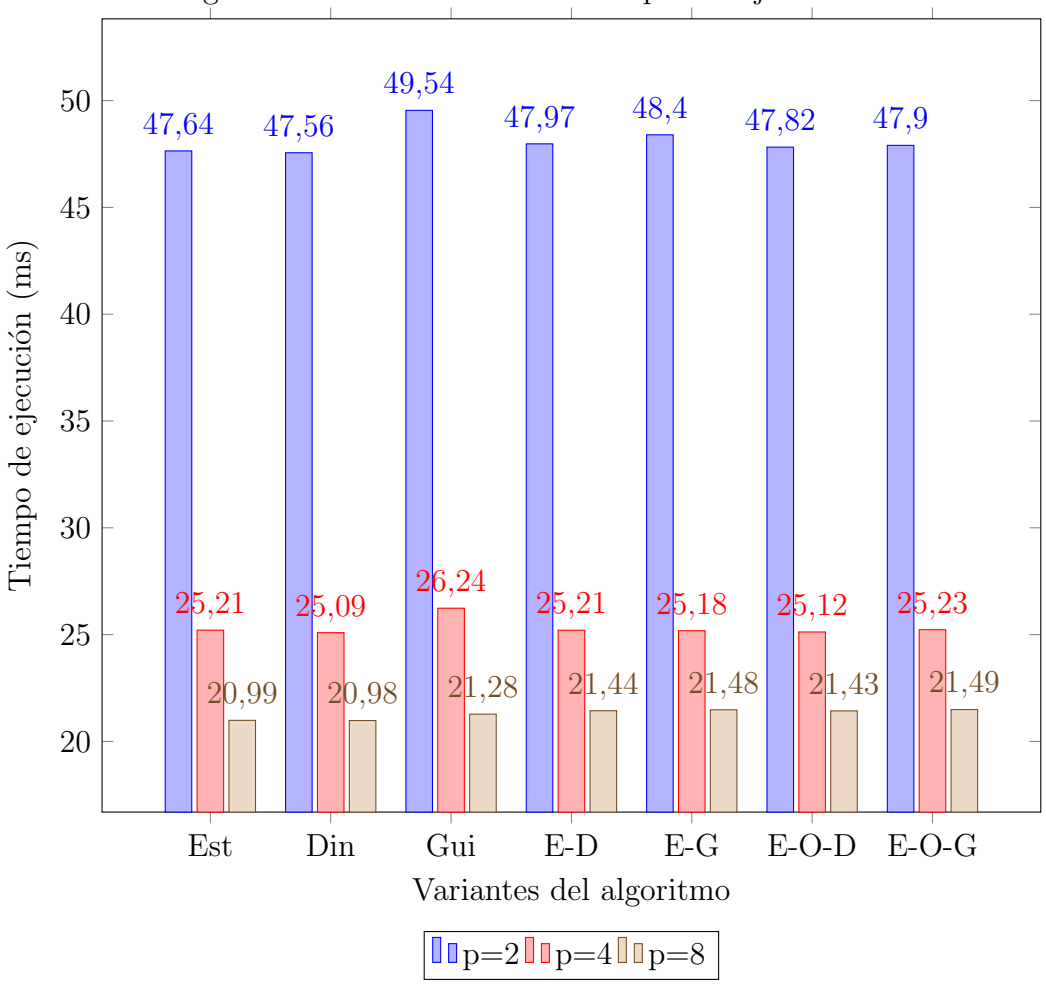
La Tabla 3.8 resume los mejores tiempos de ejecución obtenidos para cada una de las formas de asignación de tareas. Se puede observar que las particiones de *chunksizes 1* y *fila/p* favorecen el balance de carga entre los procesadores. También se evidencia que el balance de carga a través del ordenamiento de los datos se compensa con la sobrecarga producida por los cómputos requeridos al ordenar los datos. El menor tiempo de ejecución para todas las combinaciones de procesadores corresponde a un balance dinámico, en su mayoría utilizando particiones de *chunksize fila/p*. Sustentado en el análisis anterior, se propone la variante dinámica con *chunksize fila/p* como resultado de la investigación.

Tabla 3.8: Resumen de las mediciones de los tiempos de ejecución del algoritmo.

Asignación	$p = 2$		$p = 4$		$p = 8$	
	Chunksize	**t**	**Chunksize**	**t**	**Chunksize**	**t**
Estática	1	47,6441	1	25,2089	1	20,9871
Dinámica	fila/p	47,5577	1	25,0905	fila/p	20,9234
Guiada	fila	49,5427	fila	26,2356	fila/p	21,271
Estática-Dinámica	1	47,974	1	25,2059	1	21,4364
Estática-Guiada	nivel	48,397	fila/p	25,1817	fila/p	21,4787
Estática-Ord-Dinámica	fila/p	47,8202	1	25,1233	fila/p	21,4053
Estática-Ord-Guiada	1	47,904	fila/p	25,2292	1	21,4345

En la Figura 3.2 se grafican los tiempos promedio de las combinaciones de asignaciones de tareas y *chunksizes* que menores tiempos arrojaron. Gráficamente se identifican algunas diferencias entre las variantes analizadas al ejecutarse sobre diferentes cantidades de procesadores. Por ejemplo, si bien la distribución guiada ocupa valores altos en el caso de 2 y 4 procesadores, al utilizar 8 procesdores los tiempos se reducen drásticamente, arrojando uno de los mejores resultados obtenidos.

Figura 3.2: Mediciones de los tiempos de ejecución.



Para poder realizar un análisis más completo, se triangulan los resultados obtenidos para 2, 4 y 8 procesadores. La triangulación básicamente consistió en ordenar de forma ascendente las variantes para 2, 4 y 8 procesadores respectivamente, luego a cada variante se le asigna un valor que indica la posición que ocupa. Finalmente para cada variante se determina el promedio considerando los valores asignados en las tres combinaciones de procesadores. Los valores más pequeños indican menores tiempos de ejecución, mientras que los valores más altos corresponden a mayores tiempos, por lo que no se consideran soluciones adecuadas.

La Tabla 3.9 muestra los resultados obtenidos en la tringulación. Como puede apreciarse en esta tabla las variantes *estática* y *estática-ordenamiento-dinámica* tienen comportamientos similares, al igual que las variantes *guiada* y *estática-ordenamiento-guiada*. Para mayor tamaño de entrada es posible que la variante *estática-ordenamiento-dinámica* se comporte similar o mejor que la variante *estática*. Por otra parte, mientras las variantes *guiada* y *estática-ordenamiento-guiada* arrojaron los peores resultados, la variante *dinámica* se posi-

ciona como la más apropiada, seguida de la variante *estática* con una diferencia de 2.66.

Tabla 3.9: Triangulación de los resultados obtenidos para 2, 4 y 8 procesadores

Posición	$p = 2$	$p = 4$	$p = 4$	**Resultado**	
1	Dinámica	Dinámica	Dinámica	Dinámica	3/3
2	Estática	Est-Ord-Din	Estática	Estática	8/3
3	Est-Ord-Din	Est-Gui	Guiada	Est-Ord-Din	9/3
4	Est-Ord-Gui	Estática	Est-Ord-Din	Est-Din	15/3
5	Est-Din	Est-Din	Est-Din	Est-Gui	15/3
6	Est-Gui	Est-Ord-Gui	Est-Gui	Est-Ord-Gui	17/3
7	Guiada	Guiada	Est-Ord-Gui	Guiada	17/3

3.3 Evaluación experimental del algoritmo

Se plantea que un programa paralelo debe correr aproximadamente a la misma velocidad que el programa secuencial cuando solo hay un núcleo disponible. En la Tabla 3.10 se comparan los tiempos obtenidos por la variante secuencial y la variante paralela al ser aplicadas al bloque experimental O48, pero esta vez utilizando de 2 a 200 ensayos como restricción de la vecindad de búsqueda. Se observan tiempos similares para ambas soluciones, con una diferencia de solo 0.02 s al comparar los tiempos promedios de ejecución. Por tanto, se considera que el algoritmo paralelo propuesto no añade procesamiento adicional que afecte los tiempos, y que las pequeñas diferencias existentes se deben al azar.

Tabla 3.10: Comparación de los tiempos del algoritmo secuencial y el algoritmo paralelo usando un procesador.

Corrida	**t(secuencial)**	**t(paralelo)**
1	89,5289	89,3294
2	89,2769	89,5556
3	89,056	89,3481
4	89,5512	89,7134
5	89,3029	89,3069
6	89,6459	89,3047
7	89,3669	89,2708
8	89,3063	89,5275
9	89,7232	89,3263
10	89,5361	89,8446
Promedio	89,42943	89,45273
Tiempo de ejecución	89,43	89,45

Se realizó la evaluación experimental del algoritmo variando el tamaño de entrada (n) y el número de procesadores (p), atendiendo al tiempo de ejecución (t), la ganancia de velocidad (Sp) y la eficiencia (E). Se realizaron 10 corridas en cada experimento, descartándose los valores atípicos a través del método de los cuartiles.

Se utilizaron 4 juegos de datos generados aleatoriamente variando únicamente la cantidad de bloques, el resto de los parámetros de entrada se mantuvieron constantes. Esta decisión está fundamentada en que la escalabilidad del algoritmo depende directamente de la cantidad de bloques. Los juegos de datos estaban compuestos por 5000 bloques y 20000 ensayos, 10000 bloques y 20000 ensayos, 20000 bloques y 20000 ensayos, 40000 bloques y 20000 ensayos, respectivamente.

Se utilizó una vecindad esférica de 25 metros de radio, y como restricciones de la vecindad, 2 ensayos como mínimo y 200 ensayos como máximo, así como una búsqueda por octantes. El proceso de Krigeado se realizó por bloques, considerando un nivel de discretización de 5x5x5. Se utilizó un modelo de variograma anidado compuesto por 2 estructuras: una estructura esférica y otra exponencial, con 0.05 como valor de pepita, y 0.03 y 0.30 como valores de meseta, respectivamente.

Las Tabla 3.11 contiene un resumen de los resultados de los experimentos realizados para cada juego de datos usando 1, 2, 4 y 8 procesadores.

Tabla 3.11: Tiempos de ejecución de los experimentos realizados.

Ensayos	**Bloques**	**Discretización**	**Vecinos**	**p=1**	**p=2**	**p=4**	**p=8**
20000	5000	5x5x5	2:200	68,5212	37,3708	19,7337	17,4625
20000	10000	5x5x5	2:200	125,5452	68,9043	36,7482	33,1883
20000	20000	5x5x5	2:200	234,2342	128,9978	67,8316	63,6280
20000	40000	5x5x5	2:200	361,6576	203,883	107,8249	100,5069

Los datos arrojados por la evaluación experimental del algoritmo evidencian una notable disminución de los tiempos requeridos en la interpolación de Krigeado Ordinario. La mayor ganancia de velocidad (3.9239) se obtuvo para un tamaño de entrada de 20000 ensayos y 5000 bloques, donde se logró disminuir aproximadamente 4 veces el tiempo de ejecución de 68,5212 s a 17,4624 s al utilizar 8 procesadores.

Se puede observar que para un mismo tamaño de entrada, se incrementa la ganancia de velocidad al incrementar el número de procesadores, tendiendo a su valor óptimo al utilizar 2 procesadores. Al incrementar los tamaños de entrada, para un mismo número de procesadores la ganancia de velocidad disminuye ligeramente. Al utilizar 8 procesadores no se produce una ganancia de velocidad significativa, al compararse con la ganancia de velocidad obtenida para

4 procesadores.

Figura 3.3: Ganancia de velocidad del algoritmo en función del tamaño de entrada.

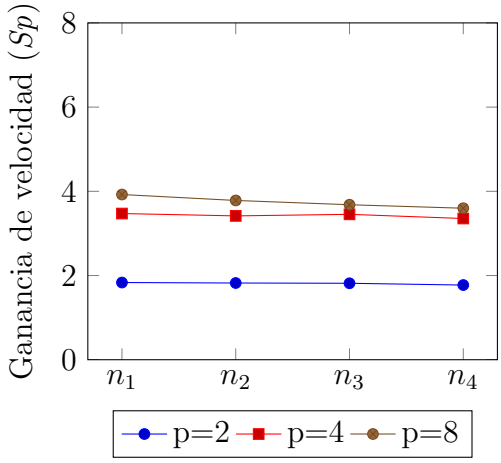
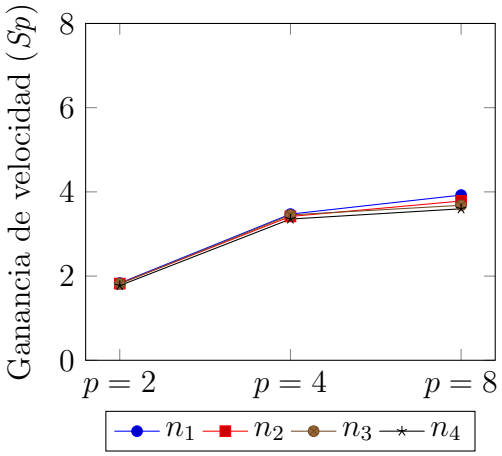


Figura 3.4: Ganancia de velocidad del algoritmo en función del número de procesadores.



Por otro lado, los valores de eficiencia indican un aprovechamiento del 88.59 % al 91.68 % de las capacidades de procesamiento al utilizar 2 procesadores, del 83.85 % al 88.61 % en el caso de 4 procesadores y un aprovechamiento cercano a la mitad al utilizar 8 procesadores. La mayor eficiencia corresponde a 91.68 % y se obtuvo para un tamaño de 5000 bloques y 20000 ensayos, al utilizar solo 2 procesadores. Se puede observar que para un mismo tamaño de entrada la eficiencia disminuye al incrementar el número de procesadores. También se observa que para mantener una eficiencia por encima del 75 % deben utilizarse 4 procesadores como máximo.

El análisis de escalabilidad para tamaños escalables arroja un decrecimiento notable de la eficiencia al incrementar simultáneamente el tamaño de entrada y el número de procesadores;

Figura 3.5: Eficiencia del algoritmo (%) en función del tamaño de entrada.

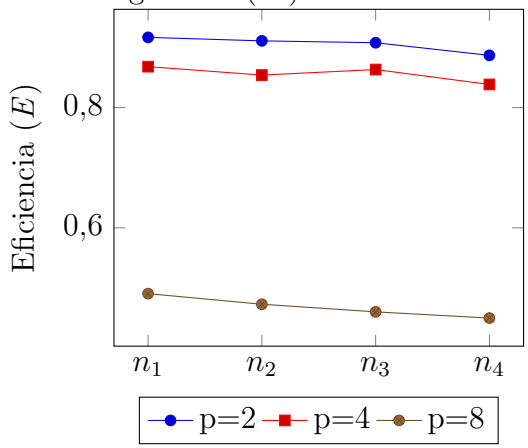
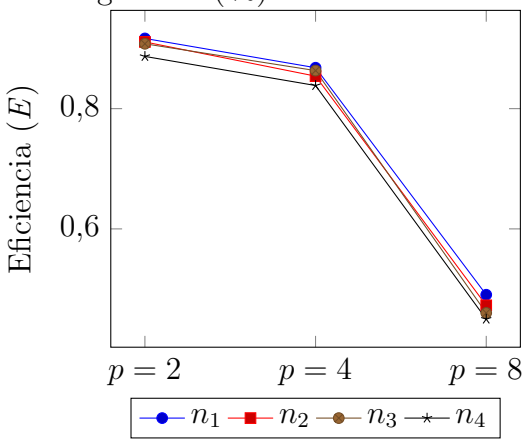


Figura 3.6: Eficiencia del algoritmo (%) en función del número de procesadores.



y que la ganancia de velocidad para 8 procesadores merma considerablemente, alejándose del número de procesadores. Estos elementos permiten concluir que el algoritmo se caracteriza por una escalabilidad pobre.

Tabla 3.12: Análisis de escalabilidad.

Ensayos	**Bloques**	**Discretización**	**Vecinos**	***p***	***Sp***	***E***
20000	10000	5x5x5	0:200	2	1,8220	0,9110
20000	20000	5x5x5	0:200	4	3,4532	0,8633
20000	40000	5x5x5	0:200	8	3,5983	0,4498

3.4 Consideraciones sobre el uso de memoria

Para la realización de los cálculos y asignaciones internas del algoritmo, la cantidad de memoria requerida está en función del número de ensayos vecinos a utilizar en las estimaciones. El número de ensayos vecinos está limitado por el número mínimo y máximo de ensayos permitidos como restricciones de la búsqueda. Por tanto, la cota superior de la complejidad espacial está dada por el máximo número de ensayos permitidos, denotado en lo adelante por v. Este valor define los tamaños de las 3 matrices que se construyen en el proceso de interpolación: la matriz de covarianza entre ensayos (de tamaño $\theta(v^2)$), la matriz de covarianza bloque-ensayos/punto-ensayos (de tamaño $\theta(v)$), y la matriz de pesos (de tamaño $\theta(v)$). El resto de las variables son escalares, objetos y pequeñas listas cuyas dimensiones no se consideran significativas.

El almacenamiento de los datos de entrada ocupa el mayor consumo de memoria, depende principalmente del listado de ensayos y del modelo de bloques. No necesita memoria extra para la salida generada por el algoritmo, estos resultados (escalares) se almacenan directamente en el modelo de bloques, proporcionado como entrada. Por otra parte, en las operaciones de indexación y búsqueda espacial el consumo de memoria es mínimo, ya que solo se almacenan las posiciones (enteros) que ocupan los ensayos, de modo que puedan ser rápidamente accedidos, sin necesidad de repetir la información correspondiente a los ensayos.

En la práctica, los consumos de memoria según el monitor de recursos del sistema, se muestran en la Tabla 3.13:

Tabla 3.13: Resultados del consumo de memoria.

	Ensayos	Bloques	Discretización	Vecinos	Memoria(MB)	Tiempo(s)
1	20000	5000	5x5x5	0:200	17,5008	16,7667
2	20000	10000	5x5x5	0:200	32,9204	26,6667
3	20000	20000	5x5x5	0:200	63,6071	47,6
4	20000	40000	5x5x5	0:200	100,354	78,6667

3.5 Conclusiones parciales

- Las pruebas de Friedman demostraron la existencia de diferencias significativas entre las variantes analizadas en cuanto al tiempo de ejecución, identificándose la variante dinámica como la que menores tiempos ofrece.
- Los resultados prácticos demuestran que la asignación de las tareas de forma dinámica combinado con una correcta granularidad, permite una distribución más uniforme de

la carga de trabajo entre los procesadores.

- La correcta selección del tamaño de los trozos de iteraciones compensa los altos tiempos de sobrecarga en las asignaciones de tareas de forma dinámica, contribuyendo directamente a equilibrar la carga de trabajo entre los procesadores.
- Los resultados prácticos demostraron que el algoritmo propuesto es aproximadamente 4 veces más rápido, que su versión secuencial siempre que se use, para su ejecución, un sistema paralelo dedicado con 8 procesadores.
- Las pruebas realizadas al algoritmo propuesto para la interpolación de Krigeado Ordinario, mostraron que el algoritmo presenta una efectividad superior al 75 % al utilizar como máximo 4 procesadores.
- El análisis de la complejidad espacial del algoritmo de Krigeado Ordinario, arrojó que los datos de entrada ocupan el mayor consumo, mientras que para los cálculos y asignaciones internas del algoritmo se estimó un consumo de memoria principal expresado en $O(v^2)$.

Conclusiones

- La evaluación experimental del algoritmo de Krigeado Ordinario demostró que la asignación de las tareas de forma dinámica combinado con una granularidad de tamaño *fila/p*, permite una distribución más equilibraba de la carga de trabajo entre los procesadores.
- Los experimentos demostraron que el algoritmo propuesto es aproximadamente 4 veces más rápido, que su versión secuencial siempre que se use, para su ejecución, un sistema paralelo dedicado con 8 procesadores.
- El algoritmo propuesto para la interpolación de Krigeado Ordinario, ofrece menores tiempos en los procesos de estimación de reservas de minerales útiles, a partir del aprovechamiento eficiente de los recursos computacionales, mediante técnicas de programación paralela en memoria compartida y el uso de métodos optimizados de búsqueda espacial.
- La reducción de los tiempos asociados a la interpolación de Krigeado Ordinario soportará la toma de decisiones rápidas, por ejemplo: durante la evaluación de la factibilidad de los proyectos mineros, así como la planificación minera de estos en condiciones de rentabilidad económica.

Recomendaciones

- Aplicar la estrategia de paralelización propuesta al resto de las variantes de interpolación espacial de Krigeado, que permitan resolver problemas que satisfacen los supuestos matemáticos apropiados en tiempos razonables, fundamentalmente en el campo de las Geociencias.
- Incluir la rotación de matrices, para el algoritmo propuesto sea aplicable a situaciones en las que el elipsoide de búsqueda y/o las estructuras de variogramas se encuentran rotadas con respecto a uno o varios ejes de coordenadas.
- Explotar la computación general en unidades de procesamiento gráfico para lograr la aceleración de la interpolación de Krigeado, aprovechando la optimización de operaciones matriciales, y las posibilidades de procesamiento paralelo masivo de estos dispositivos.

Apéndice A.
Operacionalización de las variables

		Definición operacional	
Variable	**Tipo de variable**	**Indicadores**	**Unidad medida**
Algoritmo de interpolación de krigeado ordinario	(Variable independiente) Cualitativa ordinal	Grado de paralelismo: medida de la divisibilidad del algoritmo en partes independientes de igual coste computacional.	Alto, Medio, Bajo
		Ganancia de velocidad (Sp): incremento de la velocidad del algoritmo utilizando múltiples procesadores.	Número positivo
		Eficiencia (Ep): Grado de utilización de un sistema multiprocesador.	Porciento
		Escalabilidad: Capacidad de incrementar la ganancia de velocidad en proporción al número de procesadores.	Escalable, No escalable
		Costo (Cp): Relación entre el tiempo secuencial óptimo y el tiempo paralelo multiplicado por el número de procesadores.	Costo óptimo, Costo alto, Costo medio, Costo bajo
Tiempo de ejecución del algoritmo	(Variable dependiente) Cuantitativa continua	Complejidad temporal: función T(n) que mide el número de instrucciones realizadas por el algoritmo para procesar n elementos de entrada.	Cantidad de instrucciones
		Tiempo paralelo (Tp): tiempo que tarda el algoritmo en ejecutarse en p procesadores.	Milisegundos

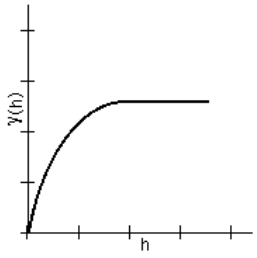
Apéndice B.
Métodos geoestadísticos de interpolación espacial

Tabla 14: Métodos geoestadísticos de interpolación espacial. Tomado de (Li y Heap, 2008)

Univariada	**Multivariada**	**Métodos combinados**
Kriging simple	Kriging universal	Análisis de tendencia de superficies combinado con Kriging
Kriging Ordinario	Kriging simple con variación de medias locales	Gradiente combinado con Kriging
Kriging de bloques	Kriging con deriva externa	Arboles de regresión combinado con Kriging
Kriging factorial	Cokriging simple	Kriging de regresión
Kriging dual	Cokriging Ordinario	
Kriging indicador	Cokriging Ordinario estandarizado	
Kriging disjuntivo	Kriging de componente principal	
Kriging basado en modelo	Cokriging colocado	
Simulación	Kriging dentro de estratos	
	Kriging factorial multivariado	
	krigin indicador	
	Cokriging indicador	
	Krigig de probabilidad	
	Simulación	

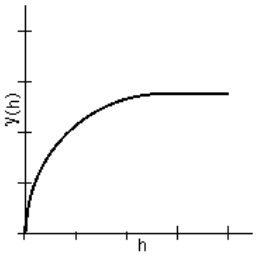
Apéndice C.
Modelos téoricos de variograma

Modelo esférico



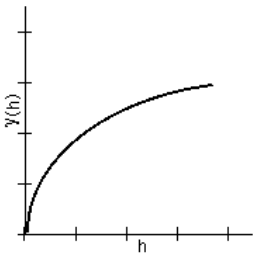
$$\gamma(h) = c_0 + c\left(\frac{3h}{2a} - \frac{1}{2}\left(\frac{h}{a}\right)^3\right) \qquad 0 < h \leq a$$
$$\gamma(h) = c_0 + c \qquad h > a$$
$$\gamma(h) = 0$$

Modelo circular



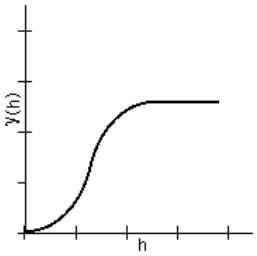
$$\gamma(h) = c_0 + c\left(1 - \frac{2}{\pi}\cos^{-1}\left(\frac{h}{a}\right) + \sqrt{1 - \frac{h^2}{a^2}}\right)$$
$$0 < h \leq a$$
$$\gamma(h) = c_0 + c \qquad h > a$$
$$\gamma(h) = 0$$

Modelo exponencial



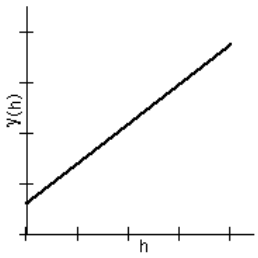
$$\gamma(h) = c_0 + c\left(1 - exp\left(-\frac{h}{r}\right)\right) \qquad h > 0$$
$$\gamma(h) = 0$$

Modelo gaussiano



$$\gamma(h) = c_0 + c\left(1 - exp\left(\frac{h^2}{r^2}\right)\right) \qquad h > 0$$
$$\gamma(h) = 0$$

Modelo lineal



$$\gamma(h) = c_0 + c\left(\frac{h}{a}\right) \qquad 0 < h \leq a$$
$$\gamma(h) = c_0 + c \qquad h > a$$
$$\gamma(h) = 0$$

Bibliografía

Estructuras de datos y algoritmos. Usando C, cap. Complejidad temporal. Universidad Técnica Federico Santa María, Valparaíso Chile, 2009.

AI-GEOSTATS. Tools for analysing geoestatistical data. online: https://wiki.52north.org/bin/view/AI-GEOSTATS/SoftwareGeostatistical, 2011.

M. A. Alfaro. *Estimación de recursos mineros*. 2007.

V. Allombert, D. Michea, F. Dupros, C. Bellier, B. Bourgine, H. Aochi, y S. Jubertie. An out-of-core gpu approach for accelerating geostatistical interpolation. En Elsevier, ed., *Procedia Computer Science. (ICCS 2014. 14th International Conference on Computational Science)*, tomo 29, págs. 888–896. 2014.

M. Armstrong y J. Carignan. *Géostatistique Linéaire, Application au Domaine Minier*. École de Mines de Paris, 1997.

C. Campbell y A. Miller. *Parallel programming with Microsoft Visual C++. Design patterns for decomposition and coordination on multicore architectures*. ISBN 978-0-7356-5175-3. Microsoft, 2011.

B. Chapman, G. Jost, y R. van der Pas. *Using OpenMP Portable Shared Memory Parallel Programming*. ISBN-13: 978-0-262-53302-7. The MIT Press, Cambridge, Massachusetts, London, England, 2008.

T. Cheng. Accelerating universal kriging interpolation algorithm usind cuda-enabled gpu. *Computers & Geosciences*, págs. 178–183, 2013.

T. Cheng, D. Li, y Q. Wang. On parallelizing universal kriging interpolation based on openmp. En Guo Quingping y Guo Yucheng, eds., *Proceedings of The Ninth International Symposium on Distributed Computing and Applications to Business Engineering and Science*, ISBN 978-0-7695-4110-5, págs. 36–39. The Institute of Electrical and Electroics Engineers, Inc., IEEE Computer Society, Hong Kong, China, 2010.

M. Chica. *Análisis Geoestadístico en el Estudio de la Explotación de Recursos Minerales*. Tesis Doctoral, Universidad de Granada, 1997.

J. Q. Cuador. *Estudios de estimación y simulación geoestadística para la caracterización de parámetros geólogo industriales en el yacimiento laterítico Punta Gorda*. Tesis Doctoral, Universidad de Pinar del Río, 2002.

J. Deraisme y Ch. De Fouquet. The geostatistical approach for reserves. *Minig Magazine*, 1996.

C. Deutsch y A. Journel. *GSLIB: Geostatistical Software Library and User's Guide (Applied Geostatistics)*. Applied Geostatistics, ISBN-10: 0195100158, ISBN-13: 978-0195100150. Oxford University Press, 2 ed$^{\underline{\text{ón}}}$., 1997.

M. A. Díaz. *Geoestadística Aplicada*. 2002.

ESRI. Arcgis resources. online: http://resources.arcgis.com, 2012.

E. Estévez. Apuntes sobre estimación de recursos y reservas. 2002.

M. J. Flynn. Some computer organizations and their effectiveness. *IEEE Transactions on Computers*, 21(9):948–960, 1972.

I. Foster. *Designing and Building Parallel Programs*. ISBN:0201575949. Addison Wesley, 1995.

A. Gallardo y F. T. Maestre. *Introducción al análisis espacial de datos en ecología y ciencias ambientales. Métodos y aplicaciones*, cap. Métodos geoestadísticos para el análisis de datos ecológicos espacialmente distribuidos, págs. 216–272. ISBN 978-84-9849-308-5. España, 1 ed$^{\underline{\text{ón}}}$., 2008.

P. M. Gausellino, E. Santos, J, y I. Zyserman, Fabio. Análisis del comportamiento de algoritmos paralelos de elementos finitos con descomposición de dominio en aplicaciones geofísicas. *Mecánica Computacional*, 20:474–481, 2001.

Geovariances. Isatis, all-in-one software for geostatistics. online: http://www.geovariances.com/en/isatis-all-in-one-software-for-geostatistics-ru324, 2016.

I. M. Glacken y D. V. Snowden. *Mineral Resource and Ore Reserve Estimation The AusIMM Guide to Good Practice*, cap. Mineral Resource Esimation, págs. 189–198. The Australian Institute of Mining and Metallurgy (AusIMM), 2001.

F. J. Gomariz. Distribución espacial: interpolación espacial. online, 2013.

P. Goovaerts. Geostatistical approaches for incorporating elevation into the spatial interpolation of rainfa. En *Journal of Hidrology*, págs. 113–129. 2000.

A. Grama, A. Gupta, G. Karypis, y V. Kumar. *Introduction to Parallel Computing*. ISBN 0-201-64865-2. Addison Wesley, 1 ed$^{\acute{o}n}$., 2003.

E. Gutiérrez de Ravé, F. J. Jiménez, A. B. Ariza, y J. M. Gómez. Using general-purpose computing on graphics processing units (gpgpu) to accelerate the ordinary kriging algorithm. *Computers & Geosciences*, (64):1–6, 2014.

M. Hessami, F. Anctil, y A. A. Viau. Delaunay implementation to improve kriging computing efficiency. *Computers & Geociences*, (27):237–240, 2001.

S. J. Jeffrey, J. O. Carter, K. B. Moodie, y A. R. Beswick. Using spatial interpolation to construct a comprehensive archive of australian climate data. *Environmental Modelling & Software*, 16(4):309–330, 2001.

K. E. Kerry y K. A. Hawick. Kriging interpolation on high-performace computers. Inf. Téc. TR: DHPC-035, Department of Computer Science, Universidad de Adelaide, 1998.

J. P Kleijnen. Kriging metamodeling in simulation: a review. *European Journal of Operational Research*, 192(3):707–716, 2009.

G. R. Lagos. Estudio de métodos de optimización robusta para el problema de planificación de producción en minería a cielo abierto. 2011.

G. Le. Multivariate geostatistics, cfsg course. Centro de Geoestadística, Escuela de Minas de Paris, 2005.

J. Li y A.D. Heap. A review of spatial interpolation methods for environmental scientists. Inf. Téc. GeoCat No. 68229, Geosciences Australia, 2008.

C. D. Lloyd. *Local models for spatials analysis*. ISBN 9780415316811. First Edition CRC Press, 2006.

K. Lu. Grass-based high perfomance spatial interpolation component for spatial decision support systems. En *Proceedings of the FOSS/GRASS Users Conference*, págs. 1–13. Bangok, Thailand, 2004.

A. Martínez y J. Ramirez. Desarrollo actual de la geoestadística en el mundo. *Minería & Geología. Revista de Ciencias de la Tierra*, 21(4):1–21, 2005.

G. Matheron y W. J. Kleingeld. The evolution of geostatistics. En *Twentieth International Symposium on the Application of Computers and Mathematics in the Mineral Industries*, tomo 3. The Southern African Institute of Mining and Metallurgy, 1987.

M. McCool, A. Robison, y J. Reinders. *Structured Parallel Programming*. ISBN 978-0-124-15993-8. Morgan Kaufmann, 2012.

J. M. Montero y L. B. Larraz. *Introducción a la geoestadística lineal*. ISBN 978-84-9745-347-9. NetBiblo S.L, La Coruña, España, 1 ed$^{\underline{ón}}$., 2008.

L. Muller. Evaluación del rendimiento de algoritmos paralelos y/o concurrentes. 2011.

D. Murillo, I. Ortega, J. D. Carrillo, A. Pardo, y J. Rendón. Comparación de métodos de interpolación para la generación de mapas de ruido en entornos urbanos. *Ing. USBMed*, 3(1), 2012.

R. M. Naiouf. *Procesamiento paralelo. Balance de carga dinámico en algortimos de sorting*. Tesis Doctoral, Universidad Nacional de La Plata, 2004.

R. Olea. *Geostatistical glossary and multilingual dictionary*. Oxford University Press, New York, 1991.

E. Olinda y G. Hernández. Un enfoque propuesto para las búsquedas por rangos. Inf. téc., Proyecto de la UPM, 2002.

C. A. Osorio. *Parallelization of Web Services and Cloud Computing: A case study of geostadistical methods*. Proyecto Fin de Carrera, Master Programs in Geoespatial Technologies, Castellon de la Plana, 2011.

A. Pesquer, Ll.; Cortés y X. Pons. Parallel ordinary kriging interpolation incorporating automatic variogram fitting. *Computers and Geosciences*, págs. 464–473, 2011.

R. E. Peña y A. Legrá. Nuevo enfoque al problema de la optimización de redes de exploración en yacimientos lateríticos. En *I Convención Cubana de Ciencias de la Tierra*. Sociedad Cubana de Geología, 2005.

E. D. Quesada. *Algoritmo para el cálculo del tonelaje en modelos de bloques con restricciones geométricas*. Proyecto Fin de Carrera, Universidad de las Ciencias Informáticas, La Habana, 2014.

N. Remy, A. Boucher, y J. Wu. *Applied Geostatistics with SGeMS: A User's Guide*. ISBN-10: 0521514142, ISBN-13: 978-0521514149. Cambridge University Press, 1 ed$^{\underline{\text{ón}}}$., 2009.

E. R. Richard, L. D. Jennifer, y L. B. Louisa. Kriging in the shadows: Geostatistical interpolation for remote sensing. *Remote Sensing of Environment*, 49(1):32–40, 1994.

V. M. Sanz. *Análisis de rendimiento de un algoritmo de diagonalización de matrices por el método por el método de Jacobi sobre una arquitectura multicore*. Proyecto Fin de Carrera, Universidad Nacional de La Plata, 2012.

D. Sullivan y D. Unwin. *Geographic Information Analysis*. John Wiley - Sons Hoboken, 2002.

S. Toub. *Patterns of Parallel Programming. Understanding and Applying Parallel Patterns with the .NET Framework 4 and Visual C#*. Microsoft Corporation, 2010.

R. Vasan, B.; Duraiswami y R. Murtugudde. Efficient kriging for real-time spatio-temporal interpolation. En *20th Conference on Probability and Statistics in the Atmospheric Sciences*. 2010.

R. Webster y M. A. Oliver. *Geostatistics for Environmental Scientists*. ISBN-10: 0470028580, ISBN-13: 978-0470028582. Wiley, 2 ed$^{\underline{\text{ón}}}$., 2007.