Ciencias Matemáticas, Vol. 32, No. 2, Pag. 141-147, 2018
Recibido 02-2018

Algoritmo Optimizado para la Interpolación Espacial del Krigeado Ordinario
Optimized Algorithm for Spatial Interpolation of Ordinary Kriging

Milenis Fernández Díaz[1]*, José Quintín Cuador Gil[2], César Rául García Jacas[3]

Resumen Los métodos de interpolación espacial proporcionan herramientas para la estimación de valores en localizaciones no muestreadas utilizando las observaciones cercanas. La interpolación de Krigeado Ordinario es uno de los métodos geoestadísticos más frecuentemente usados para la realización de análisis espaciales. 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. La reducción del tiempo de ejecución de aplicaciones de interpolación espacial puede ser un objetivo de alta prioridad, por ejemplo, en sistemas que soportan la toma de decisiones rápidas. Con el objetivo de disminuir los tiempos asociados a la interpolación espacial del 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 que satisfacen los supuestos matemáticos apropiados 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 algebra lineal optimizada para los cálculos matriciales. El algoritmo propuesto permite una mayor rapidez en la interpolación espacial de Krigeado Ordinario, logrando un mejor aprovechamiento de los recursos de cómputo instalados.

Abstract Spatial interpolation methods provide tools for estimating values at unsampled locations using nearby observations. Ordinary Kriging interpolation is one of the most frequently used geostatistical methods for performing spatial analysis. Its objective is find the Best Linear Unbiased Estimator from the data available, which generally are insufficient because of the cost of obtaining it. It was characterized by expensive linear algebra operations affecting high runtimes fundamentally solve large systems of linear equations. Reducting the runtime of spatial interpolation applications can be a high priority target, for example, in systems that support quick decisions. In order to reduce the time associated to spatial interpolation of Ordinary Kriging, it was proposed an algorithm based on the use of parallel programming techniques and optimized search methods; for resolving problems meeting the appropriate mathematical assumptions at reasonable time, mainly in the field of Geosciences. This algorithm was implemented using C++11 programming language, OpenMP 4.8.2 as library of shared memory parallel programming, and Atlas CLAPACK as linear algebra library optimized for matrix calculations. The proposed algorithm allows faster in the spatial interpolation of Ordinary Kriging, making better use of computing resources installed.

Palabras Clave
geoestadística — interpolación espacial — Krigeado Ordinario — programación paralela

[1] *Centro de Geoinformática y Señales Digitales, Universidad de las Ciencias Informáticas, La Habana, Cuba, mfdiaz@uci.cu*
[2] *Departamento de Física, Universidad de Pinar del Río, Pinar del Río, Cuba, cuador@upr.edu.cu*
[3] *Centro de Estudios de Matemática Computacional, Universidad de las Ciencias Informáticas, La Habana, Cuba, crjacas@uci.cu*
***Autor para Correspondencia**

Introducción

La interpolación espacial de Krigeado tiene como objetivo encontrar el Mejor Estimador Lineal Insesgado a partir de los datos disponibles [3], los cuales generalmente son insuficientes debido al costo de su obtención. Se basa en el principio de que las variables espaciales de una determinada población se encuentran correlacionadas en el espacio; es decir que mientras más cercanos estén dos puntos sobre la superficie terrestre,

menor será la variación de los atributos medidos [4]. 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 constituye un método de interpolación espacial muy utilizado en la construcción de superficies y 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. Precisamente, es en el campo de la Minería, donde se origina el Krigeado de manos del ingeniero en minas Danie Krige, al explotar la correlación espacial para hacer predicciones en la evaluación de reservas de las minas de oro en Sudáfrica [5]; y gracias a la formulación matemática de George Matheron en la Escuela de Minas de París.

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: [7, 8, 11, 12, 13]. 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 observaciones disponibles y M el número de puntos a interpolar [13]. Cuando $M \approx N$, la complejidad computacional puede considerarse $O(N^4)$ lo cual no es favorable si se trabaja con grandes volúmenes de datos.

1. Métodos

1.1 Formulación matemática del krigeado ordinario

El problema del Krigeado Ordinario consiste en estimar el valor en el sitio desconocido, 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 y Krigeado de bloques. La estimación se calcula mediante la Ecuación 1:

$$z*_k = \lambda_1 z(x_1) + \lambda_2 z(x_2) + ... + \lambda_n z(x_n) \tag{1}$$

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 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 [1] :

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. $\sigma_{E^2} = var[z^*_k - z_k]$ 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 uno de los puntos se calculan mediante la resolución de este sistema de ecuaciones (Ecuación 2) [1].

$$\begin{cases} \sum_{j=1}^{n} \lambda_j \gamma(u_i - u_j) + \mu = \gamma(u - u_i) & i = 1...n \\ \sum_{j=1}^{n} \lambda_j = 1 \end{cases} \tag{2}$$

El sistema de ecuaciones también puede ser expresado en forma matricial en función de la covarianza (Ecuación 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.

$$\begin{pmatrix} \gamma(u_1 - u_1) & \cdots & \gamma(u_1 - u_n) & 1 \\ \vdots & \ddots & \vdots & \vdots \\ \gamma(u_n - u_1) & \cdots & \gamma(u_n - u_n) & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_n \\ \mu \end{pmatrix} = \begin{pmatrix} \gamma(u_n - u) \\ \vdots \\ \gamma(u_n - u) \\ 1 \end{pmatrix} \tag{3}$$

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 [2]. La varianza del error puede calcularse mediante la Ecuación 4:

$$\sigma^2_{KO} = \sum_{i=1}^{n} \lambda_i \gamma(u_i - u) - \sigma^2 - \mu \tag{4}$$

1.2 Descripción del Krigeado Ordinario

El proceso de Krigeado involucra 4 pasos fundamentales: 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; 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.

El algoritmo de Krigeado 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 la Expresión 5 (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).

$$p = \left\{(x_0, y_0, z_0), (nb_i * nb_j * nb_k), (lb_i * lb_j * lb_k), (db_i * db_j * db_k)\right\} \tag{5}$$

- $S = \left\{(R_x, R_y, R_z), (\alpha, \beta, \theta), (r_1, r_2, \cdots, r_n)\right\}$ 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_n); representada por la Ecuación 6: 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.

$$\frac{(x-x_0)^2}{a^2} + \frac{(y-y_0)^2}{b^2} + \frac{(z-z_0)^2}{c^2} = 1 \tag{6}$$

- $\gamma(h) = \{nst, c_0, (c_1\gamma_1(h) \cdots c_n\gamma_n(h))\}$ 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 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 [10]. 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.

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 coordenadas 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 7):

$$x_i = \frac{(x-x_0)}{nb_i}; y_j = \frac{(y-y_0)}{nb_j}; z_k = \frac{(z-z_0)}{nb_k}; \tag{7}$$

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 8):

$$loc = zn_xn_y + yn_x + x \tag{8}$$

1.4 Técnicas, herramientas y tecnologías

El diseño del algoritmo se basó en los siguientes principios: partición (descomposición de la computación de tareas), comunicación (coordinación en la ejecución de tareas), aglomeración (combinación de los resultados de las tareas) y mapeo (asignación de tareas a los procesadores); descritas en

[6], [9]. Se centró fundamentalmente en la partición y asignación. En el diseño del algoritmo no se presentaron secciones críticas, por lo que no se hacen copias de la lista de puntos a interpolar; todos los procesos pueden leer y escribir en esta lista sin que se generen conflictos de memoria.

El algoritmo de Krigeado Ordinario fue implementado usando C++11 como lenguaje de programación. Para los cálculos matriciales se utilizó la biblioteca de álgebra lineal Atlas CLapack, caracterizada por ser rápida. Esta biblioteca permitió simplificar el cálculo de operaciones matriciales como la resolución de sistemas de ecuaciones lineales y la multiplicación de matrices.

Las técnicas de programación paralela y distribuida han demostrado ser una alternativa viable para la solución rápida de este tipo de problemas computacionales. En la presente investigación se aplicaron técnicas de programación paralela en memoria compartida a través de la biblioteca OpenMP 4.8.2.

La biblioteca OpenMP se basa en el modelo *fork-join* para obtener el paralelismo a través de múltiples hilos. 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.

2. Resultados y discusión

2.1 Aceleración mediante OpenMP

El algoritmo propuesto para la interpolación de Krigeado Ordinario consta de 3 etapas de procesamiento paralelo a través del modelo *fork-join* (Figura 1). Este modelo plantea la división del hilo maestro en hilos esclavos que se ejecutan concurrentemente, distribuyéndose las tareas sobre estos hilos. Los hilos acceden a la misma memoria, aunque es posible gestionar estos accesos generando espacios de memoria privada. A continuación se describen las etapas de procesamiento, y se modela el algoritmo de Krigeado Ordinario en forma de pseudocódigo (Algoritmos 1 y 2).

- **Etapa I.** Se realiza la indexación espacial de los puntos en el modelo de bloques y búsqueda de vecinos a utilizar en cada una de las interpolaciones. Los bloques son distribuidos en trozos de aproximadamente igual tamaño entre los procesadores antes de que las iteraciones sean ejecutadas mediante asignaciones estáticas (*shedule static*); todas las iteraciones son repartidas de forma continua antes de ser ejecutadas.
- **Etapa II.** Se realiza el ordenamiento de los bloques a estimar en función de la cantidad de vecinos a utilizar. El modelo de bloques una vez más es particionado en tantas partes iguales como número de procesadores; luego estas partes son ordenadas simultáneamente. Una vez que concluye el proceso de ordenamiento, se mezclan los resultados arrojados por cada procesador en tantas iteraciones como sean necesarias.

Algorithm 1 Pseudocódigo del algoritmo de Krigeado Ordinario (modelo de bloques)

Require: A, M, S, $\gamma(h)$
Ensure: M
1: discretizar (M)
2: indexar (A, M)
3: $numbloques = M.cantidad$
4: {Etapa I}
5: **for** $i = 0 : numbloques$ **do**
6: $b = M.at(i)$
7: $b.vecinos =$buscar ($b.centroide$, A, M, S)
8: **end for**
9: {Etapa II}
10: ordenar (M)
11: {Etapa III}
12: **for** $i = 0 : numbloques$ **do**
13: $b = modelo.at(i)$
14: interpolar (b, A, M, S, $\gamma(h)$)
15: **end for**

Algorithm 2 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, $\gamma(h)$)
4: $matRM =$ construirMatrizDerecha (b, A, $\gamma(h)$)
5: $matR =$ resolverSEL (LM, RM)
6: $valor = 0$
7: $error = 0$
8: $varianza =$ calcularVarianzaBloque (b, A, $\gamma(h)$)
9: $\mu = R[numvecinos]$
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: $b.error = varianza - error - \mu$
16: **else**
17: $b.valor = NE$ {no estimado}
18: $b.error = NE$ {no estimado}
19: **end if**

- **Etapa III.** Se realiza la interpolación. Los bloques se distribuyen nuevamente entre los procesadores, pero esta vez de forma dinámica (*shedule dynamic*), es decir las iteraciones son asignadas de forma continua a solicitud de los procesadores, hasta que se acaben. En esta etapa se calculan los pesos y los valores asociados a los puntos a interpolación.

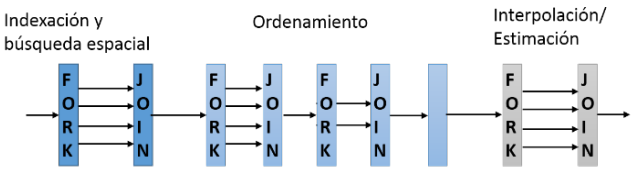


Figura 1. Utilización del modelo *fork-join* para lograr el paralelismo de datos en el algoritmo.

2.2 Experimentos y resultados

Se realizó la evaluación experimental del algoritmo variando el tamaño de entrada *(n)* y el número de procesadores *(p)*, atendiendo al tiempo de ejecución, la ganancia de velocidad (*Speed Up*) y la eficiencia *(E)*. Se entiende por tiempo de ejecución como el tiempo que transcurre desde el comienzo de la ejecución del programa en el sistema paralelo, hasta que el último procesador culmine su ejecución. A la ganancia de velocidad también se le conoce como aceleración, y consiste en la relación entre el tiempo de ejecución sobre un procesador secuencial y el tiempo de ejecución sobre múltiples procesadores. Por eficiencia se entiende el porcentaje de tiempo empleado en proceso efectivo; este indicador mide el grado de utilización de un sistema multiprocesador.

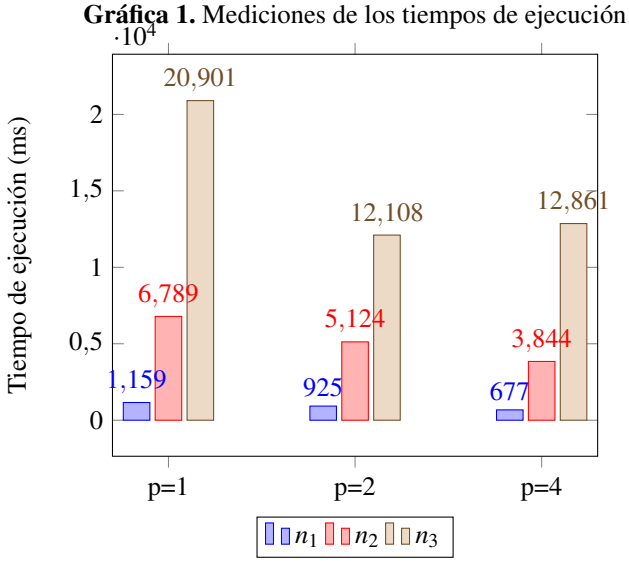
Los experimentos se ejecutaron en una estación de trabajo Acer Aspire 5755 con procesador Intel (R) Core (TM) i5-2430 (compuesta por 2 núcleos físicos y 2 virtuales para un total de 4 procesadores), con una frecuencia de 2.40 GHz, 4 GB de memoria instalada y sistema operativo Ubuntu 14.04 (32 bits). El proceso de Krigeado se realizó de forma puntual considerando los centroides de los bloques como la nube de puntos a estimar. Se realizaron 10 corridas en cada experimento, descartándose los datos atípicos a través del método de los cuartiles.

Se utilizaron 3 juegos de datos generados aleatoriamente variando el número de ensayos. Los juegos de datos estaban compuestos por 1000 bloques y 1000 ensayos (n_1), 1000 bloques y 2000 ensayos (n_2), 1000 bloques y 3000 ensayos (n_3), respectivamente. Se utilizó un modelo de variograma lineal de 1 como valor de pepita, y 10 como valor de la pendiente. Se utilizó una vecindad esférica de 5 metros de radio. Los Cuadros 1, 2 y 3 contienen los resultados de los experimentos realizados usando 1, 2 y 4 procesadores respectivamente, así como la descripción de dichos resultados a partir de estadígrafos como la mediana, cuartil menor y mayor y el promedio.

Cuadro 1. Mediciones de los tiempos de ejecución del algoritmo paralelo para un procesador (ms).

Corrida	t(n_1)	t(n_2)	t(n_3)
1	1150	6687	21023
2	1144	6820	20955
3	1152	6756	20832
4	1176	6884	20799
5	1171	6803	20884
6	1162	6809	20863
7	1155	6829	20918
8	*1215	6789	20841
9	1175	6700	20914
10	1150	6811	20985
Mediana	1158.50	6806.00	20899.00
Cuartil menor	1150	6756	20841
Cuartil mayor	1175	6820	20955
Rango intercuartil	25	64	114
Límite inferior	1112.5	6660	20670
Límite superior	1212.5	6916	21126
Promedio	1159.44	6788.8	20901.4
Tiempo de ejecución	1159	6789	20901

Gráfica 1. Mediciones de los tiempos de ejecución



Los datos arrojados por la evaluación experimental del algoritmo evidencian una disminución de los tiempos requeridos en la interpolación de Krigeado Ordinario (Gráfica 1). La mayor ganancia de velocidad (1.77) se obtuvo para un tamaño de entrada de m=1000 (bloques) y n=2000 (ensayos), donde se logró disminuir el tiempo de ejecución de 6789 ms a 3844 ms al utilizar 4 procesadores. La ganancia de velocidad incrementa al aumentar el tamaño de entrada atendiendo a su valor óptimo cuando se utilizan 2 procesadores (Gráfica 2).

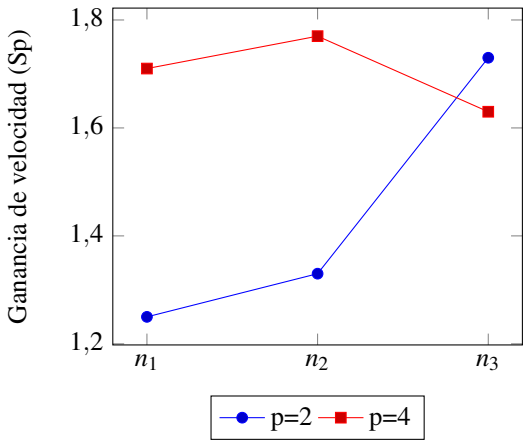
Cuadro 2. Mediciones de los tiempos de ejecución del algoritmo paralelo para 2 procesadores (ms).

Corrida	t(n_1)	t(n_2)	t(n_3)
1	721	4053	12094
2	1190	4227	*12799
3	786	4033	12064
4	760	4081	12093
5	1136	4700	4700
6	1176	6193	12039
7	757	6404	12158
8	1189	4020	12135
9	766	7085	12144
10	772	6445	12155
Mediana	779.00	4463.50	12114.50
Cuartil menor	760	4053	12087
Cuartil mayor	1176	6404	12155
Rango intercuartil	416	2351	68
Límite inferior	136	526.5	11985
Límite superior	1800	9930.5	12257
Promedio	925.3	5124.1	12107.66
Tiempo de ejecución	925	5124	12108
Ganancia de velocidad	1.25	1.33	1.73
Eficiencia	62.5	66.5	86.5

Cuadro 3. Mediciones de los tiempos de ejecución del algoritmo paralelo para 4 procesadores (ms).

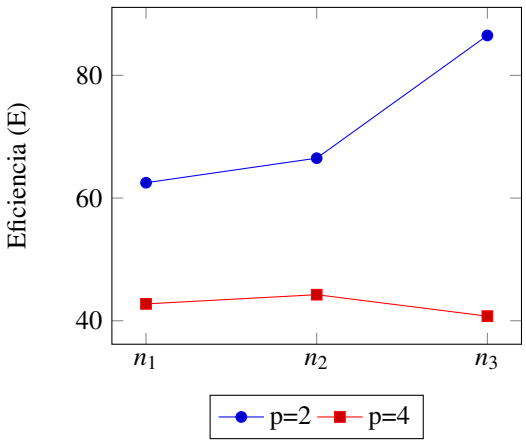
Corrida	t(n_1)	t(n_2)	t(n_3)
1	694	3781	12379
2	*757	3894	12379
3	706	3834	12437
4	672	3878	12352
5	683	3793	13196
6	685	3985	13042
7	662	3821	13434
8	664	3796	12494
9	674	3796	13188
10	657	3860	13708
Mediana	678.50	3827.50	12768.00
Cuartil menor	664	3796	12379
Cuartil mayor	694	3878	13196
Rango intercuartil	30	82	817
Límite inferior	619	3673	11153.5
Límite superior	739	4001	14421.5
Promedio	677.44	3843.8	12860.9
Tiempo de ejecución	677	3844	12861
Ganancia de velocidad	1.71	1.77	1.63
Eficiencia	42.75	44.25	40.75

Gráfica 2. Ganancia de velocidad del algoritmo.



Por otro lado, los valores de eficiencia indican un aprovechamiento del 62.5 % al 86.54 % de las capacidades de procesamiento al utilizar 2 procesadores, y un aprovechamiento cercano a la mitad al utilizar 4 procesadores. Se evidencia un notable incremento de la eficiencia al aumentar los tamaños de entrada con el uso de 2 procesadores (Gráfica 3).

Gráfica 3. Eficiencia del algoritmo (%).



Se observa una disminución de la ganancia de velocidad y la eficiencia al utilizar 4 procesadores para un tamaño de entrada de 1000 bloques y 3000 ensayos, lo cual está influenciado por el procesamiento de ordenamiento de los datos en busca de equilibrar la carga entre los procesadores. Si bien el ordenamiento por mezcla fue implementado de forma paralela, a medida que se incrementan las iteraciones se van desechando procesadores al mezclar los resultados parciales. Esto se va acentuando en la medida que aumentan los tamaños de entrada y el número de procesadores.

2.3 Conclusiones

- La complejidad de los cálculos matemáticos utilizados en la interpolación de Krigeado Ordinario, fundamental-

mente la resolución de grandes sistemas de ecuaciones, repercute en altos tiempos de ejecución que retardan los procesos de estimación.

- 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 de 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.
- 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, así como la planificación minera de estos en condiciones de rentabilidad económica.

Agradecimientos

Se agradece la colaboración del especialista José Arias de la Oficina Nacional de Recursos Minerales (ONRM).

Referencias

[1] M. A. Alfaro. *Estimación de recursos mineros*. 2007.

[2] M. Armstrong and J. Carignan. *Géostatistique Linéaire, Application au Domaine Minier*. École de Mines de Paris, 1997.

[3] M. Chica. *Análisis Geoestadístico en el Estudio de la Explotación de Recursos Minerales*. PhD thesis, Universidad de Granada, 1997.

[4] J. Deraisme and Ch. De Fouquet. The geostatistical approach for reserves. *Minig Magazine*, 1996.

[5] M. A. Díaz. *Geoestadística Aplicada*. 2002.

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

[7] K. E. Kerry and K. A. Hawick. Kriging interpolation on high-performace computers. Technical report, Department of Computer Science, Universidad de Adelaide, 1998.

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

[9] R. M. Naiouf. *Procesamiento paralelo. Balance de carga dinámico en algortimos de sorting*. PhD thesis, Universidad Nacional de La Plata, 2004.

[10] E. Olinda and G. Hernández. Un enfoque propuesto para las búsquedas por rangos. Technical report, Proyecto de la UPM, 2002.

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

[12] D. Sullivan and D. Unwin. *Geographic Information Analysis*. John Wiley - Sons Hoboken, 2002.

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