



Casa abierta al tiempo

UNIVERSIDAD AUTÓNOMA METROPOLITANA
Unidad Iztápalapa

**“Propuesta de un Sistema Paralelo para la
Creación de Redes Porosas sujetas a
Restricciones Geométricas en Arquitecturas
Multicore”**

Idónea Comunicación de Resultados que presenta:

Ing. Angel González Méndez

para obtener el grado de:

Maestro en Ciencias y Tecnologías de la Información

División de Ciencias Básicas e Ingeniería
Posgrado en Ciencias y Tecnologías de la Información

Asesora:

Dra. Graciela Román Alonso

Sinodales:

Presidente: Dr. Fernando Rojas González

Secretario: Dr. Miguel Alfonso Castro García

Vocal: Dr. Carlos Felipe Mendoza

Vocal: Dra. Graciela Román Alonso

Resumen

El estudio de los medios porosos, con el fin de entender las características específicas de los materiales o procesos capilares de los mismos, es de gran importancia para un gran número de aplicaciones industriales. El Modelo Dual de Sitios y Enlaces (MDSE) ha sido una base importante en el desarrollo de simuladores (in-silico) de medios porosos. Bajo este enfoque, se tiene que un material poroso se compone por sitios (cavidades, protuberancias) los cuales están interconectados a través de enlaces; cada sitio se interconecta con una serie de enlaces y cada enlace interconecta a dos sitios. En la actualidad, varios algoritmos computacionales para la simulación de redes porosas se han implementado, sin embargo, solo unos cuantos validan el cumplimiento de las restricciones geométricas que surgen al conectar dos enlaces adyacentes de un mismo sitio, en donde no debe existir interferencia espacial entre ellos. El validar este tipo de restricciones nos ayuda a crear redes porosas más realistas; sin embargo, la complejidad algorítmica que conlleva el cumplimiento de estas restricciones hace que el tiempo de construcción de una red aumente. En este trabajo, se parte de un algoritmo secuencial desarrollado como parte del Proyecto Interdisciplinario de los Departamentos de Ing. Eléctrica y de Química de la UAM-IZT, el cual genera redes porosas que incluyen las restricciones geométricas; el inconveniente de este algoritmo es que requiere de mucho tiempo para construir redes porosas grandes. Para solucionar dicho aspecto, se proponen dos versiones paralelas de construcción de redes porosas, validando las restricciones geométricas entre todos los poros, basándonos en el MDSE. El objetivo de la primer propuesta es paralelizar el algoritmo secuencial anteriormente desarrollado, mientras que la segunda propuesta aplica el Método de Monte Carlo paralelo en la construcción de una red porosa. Nuestras propuestas paralelas fueron implementadas bajo un modelo de memoria compartida, usando OpenMP para crear un conjunto de hilos (tareas computacionales) los cuales trabajan de forma simultánea en espacios aleatorios e independientes.

Agradecimientos

A mi madre Lucía Guadalupe Méndez González, por creer en mi en todo momento, por su apoyo incondicional a lo largo de mi vida y por ser un ejemplo de vida.

A mi padre Hilario González Estrella, a pesar de que ya no te encuentras con nosotros, tu espíritu siempre a estado conmigo.

A mi hermanas, de las cuales siempre he recibido su apoyo.

A mi tutora la Dra. Graciela Román Alonso, por su constante apoyo para la realización de este trabajo.

A los profesores del Posgrado en Ciencias y Tecnologías de la Información, por todos los conocimientos otorgados en sus clases, los cuales fueron la base para hacer posible el presente trabajo.

A la Universidad Autónoma Metropolitana - Iztapalapa, por las facilidades otorgadas para el correcto desarrollo del este trabajo y por permitirme formar parte de esta gran institución.

Al Laboratorio Supercómputo y Visualización en Paralelo, por los recursos prestados, sin los cuales esto trabajo no hubiera sido posible.

Al Consejo Nacional de Ciencia y Tecnología, por el apoyo económico otorgado a través de su programa de becas para la realización de mis estudios de posgrado.

Índice general

Resumen	III
Agradecimientos	v
Índice general	VII
Índice de figuras	IX
1. Introducción	1
2. Simulación de Redes Porosas	5
2.1. Modelado de Redes Porosas	5
2.2. Principio de Construcción	7
2.3. Restricciones Geométricas	7
3. Objetivos	9
3.1. Objetivos Particulares	9
4. Trabajo Relacionado	11
4.1. Algoritmos Secuenciales	11
4.1.1. Algoritmo BiaSED	11
4.1.2. Algoritmo NoMISS	12
4.2. Algoritmos Paralelos	14
4.2.1. Algoritmo Paralelo BiaSED	14
4.2.2. Algoritmo Paralelo S-NoMISS	15
4.2.3. Algoritmo Paralelo D-NoMISS	16
4.3. Conclusiones	17
5. Construcción Secuencial	19
5.1. Solución Aleatoria	20
5.2. Solución Híbrida	20
5.2.1. Sembrado de Clusters de sitios	21
5.2.2. Asignación de Enlaces	23
5.3. Generación de una Red Porosa válida	24
5.4. Mejoramiento de la isotropía	25

6. Construcción Paralela	29
6.1. Distribución Dinámica de la Red Porosa	29
6.2. Solución Paralela Aleatoria	31
6.2.1. Inicialización	31
6.2.2. Generación de una Red Porosa Válida	32
6.2.3. Mejoramiento de la Isotropía	32
6.3. Solución Paralela Híbrida	32
6.3.1. Inicialización	34
6.3.2. Generación de una red porosa válida	38
6.3.3. Mejoramiento de la isotropía	39
7. Resultados	41
8. Conclusiones y Trabajo Futuro	51
Referencias	53

Índice de figuras

2.1. Representación bidimensional de un material poroso mediante el MDSE	6
2.2. Red porosa con los sitios representados por esferas y los enlaces representados por cilindros, con $C = 6$	6
2.3. Traslape entre las distribuciones F_B y F_S	7
2.4. Representación del PC y RG en un sitio con dos enlaces vecinos	8
4.1. Construcción de un cluster de tamaño $3 \times 3 \times 3$ con el algoritmo NoMISS	13
4.2. Distribución de una red porosa entre 8 nodos o procesos MPI	15
5.1. Red porosa inicializada con L^3 sitios y $3 \cdot L^3$ enlaces, con $L=3$.	21
5.2. Sembrado de un cluster de sitios de tamaño $3 \times 3 \times 3$	22
5.3. Red porosa después del proceso de siembra de clusters	23
5.4. Ejemplo de un intercambio válido de dos sitios (a) selección y (b) intercambio	25
5.5. Ejemplo de un intercambio válido de dos enlaces (a) selección y (b) intercambio	26
5.6. Ejemplo de un intercambio inválido de dos sitios (a) selección y (b) intercambio	26
5.7. Ejemplo de un intercambio inválido de dos enlaces (a) selección y (b) intercambio	27
6.1. Red porosa dividida en ocho subredes; $N = 2 \cdot 2 \cdot 2$	30
6.2. Distribución de la Red para (a) 4 hilos, (b) 8 hilos, y (c) 64 hilos	30
6.3. Ejemplo de un desplazamiento lógico (a) Posición inicial $(0,0,0)$, (b) desplazamiento sobre y $(0,2,0)$	31
6.4. Cambio de origen durante el sembrado, (a) Origen en $(0, 0, 0)$ y (b) cambio de origen a $(0, L/2, 0)$	35
7.1. Tiempos de ejecución de SA y SPA utilizando 2, 8 y 64 hilos bajo diferentes valores de Ω (escala logarítmica)	42
7.2. Tiempos de ejecución de SH y SPH utilizando 2, 8 y 64 hilos bajo diferentes valores de Ω (escala logarítmica)	43
7.3. Tiempos de ejecución para SH , SPH , SA y SPA para distintos traslapes y variando el número de hilos (escala logarítmica)	44

7.4. Tiempos de ejecución para <i>SPH</i> para redes de distinto tamaño y variando el traslape (escala logarítmica)	46
7.5. Redes porosas antes y después de eliminar las violaciones a las <i>RG</i> con diferentes valores de Ω	47
7.6. Redes porosas antes y después de eliminar las violaciones a las <i>RG</i> utilizando un número diferente de hilos	48
7.7. Redes porosas antes y después de eliminar las violaciones a las <i>RG</i> de diferentes tamaños	49

Capítulo 1

Introducción

Por años se han utilizado las computadoras como herramientas para la solución, aproximación o simulación de problemas científicos reales, los cuales tienen una o varias de las siguientes características: son complejos, requieren de un gran número de cálculos o manipulan grandes volúmenes de datos. Lo anterior fue la razón por la cual nació la computación científica la cual es una actividad multidisciplinaria en la cual se hace uso de métodos y técnicas computacionales para dar soluciones exactas o aproximadas con la premisa de que al utilizar las computadoras el tiempo para obtener resultados es menor.

La simulación por computadora es una herramienta altamente utilizada debido a que nos permite obtener soluciones aproximadas sobre escenarios de la vida real, varias ramas de la ciencia se auxilian de simulaciones por computadora para lograr mejores resultados en experimentos reales basándose en la información obtenida de simulaciones. Cuando se habla de simulación por computadora, está inherente que se requiere de un modelo el cual nos ayude a abstraer características y limitaciones de un problema real. Adicionalmente de que la simulación por computadora nos puede permitir generar experimentos con mayor exactitud, ésta también nos puede ayudar a disminuir riesgos y ahorrar recursos.

En particular este proyecto hace énfasis en la simulación de materiales porosos, donde tal y como su nombre lo dice son materiales que tienen poros o huecos, estos poros son de distintos tamaños y se encuentran distribuidos a lo largo de los materiales con la peculiaridad de que dichos poros se encuentran conectados entre sí. Debido a las características que tienen los materiales porosos son altamente utilizados en la industria, al estudiar la estructura y propiedades de estos materiales y adicionalmente con la simulación por computadora se pueden encontrar nuevas aplicaciones y usos industriales.

Varios métodos para la simulación de redes porosas han sido propuestos; sin embargo, son pocos los que validan el cumplimiento de las restricciones geométricas que surgen al conectar dos o más poros adyacentes, en donde no debe existir interferencia espacial entre ellos. El validar este tipo de restricciones nos ayuda a crear redes porosas

más realistas, pero el procesamiento de millones o billones de datos que pueden representar los poros de una red, junto con la complejidad algorítmica para hacer respetar las restricciones geométricas, se vuelve un reto computacional importante.

Por lo general la computación científica hace uso del cómputo de alto rendimiento (cómputo paralelo) a través de modelos de programación paralela, los cuales permiten el uso eficiente de los recursos computacionales para disminuir el tiempo en la obtención de resultados. El cómputo paralelo fue impulsado gracias a los avances tecnológicos como lo fueron las computadoras multi-procesador o los primeros clusters (Beowulf), en ambas arquitecturas hubo un avance significativo; en la primera fue la incorporación de más de un procesador en una misma tarjeta y en la segunda fue el uso de la capacidad de cómputo de computadoras independientes a través de su interconexión (cluster). En la actualidad el cómputo paralelo va cobrando mayor fuerza y además conforme pasa tiempo se está volviendo más accesible y por lo mismo se está aplicando en un mayor número de áreas de conocimiento. En particular, para aplicar el cómputo paralelo en un problema, se requiere elegir una arquitectura y un técnica de programación. Las arquitecturas más utilizadas para el cómputo paralelo o de alto rendimiento son: multi-procesador, multi-núcleo, gpus, clusters y grids. Respecto a técnicas de programación, las podemos dividir en 3 grandes grupos: la programación con memoria compartida, la programación con memoria distribuida y/o programación híbrida.

Cuando se utilizan arquitecturas multi-núcleo, la mejor opción de programación paralela se realiza a través de la creación de hilos (tareas computacionales), para que las unidades de procesamiento puedan acceder a la memoria de manera compartida. Dos de las tecnologías más utilizadas para el manejo de hilos son: Posix Threads[13] y OpenMP[6], ambos siguen el modelo fork-join, sin embargo OpenMP proporciona una interfaz de alto nivel que permite al programador concentrar la mayor parte de su esfuerzo en resolver el problema de aplicación y no en aspectos técnicos de creación explícita de hilos y manejo de mecanismos de sincronización, que en ocasiones generan errores que no se relacionan con el problema que se quiere resolver. Además OpenMP es un estándar avalado por diversas empresas internacionales y se encuentra soportado por los principales compiladores de C/C++ y Fortran.

Este trabajo contribuye a la simulación paralela de redes porosas que consideran el cumplimiento de restricciones geométricas entre poros adyacentes, usando una arquitectura multi-núcleo y OpenMP. Se proponen dos algoritmos paralelos que permiten construir redes porosas haciendo que varios hilos de ejecución trabajen en diferentes regiones de una red porosa cúbica para disminuir el tiempo de simulación. El alcance de este proyecto se enfoca en la comparación de esas dos propuestas.

El resto de la tesis se organiza de la siguiente forma: en el Capítulo *refchamp:construction* se presentan los conceptos básicos relacionados con la simulación de las redes porosas. Posteriormente, el Capítulo 3 presenta los objetivos de este trabajo. En el Capítulo 4 se muestra en forma de resumen el trabajo relacionado en cuanto a algoritmos para la construcción de redes porosas que se basan en el MDSE y las ventajas del cómputo

paralelo sobre sistemas de memoria compartida utilizando arquitecturas multi-core. En el Capítulo 5 se presentan dos algoritmos secuenciales para la creación de redes porosas sujetas a las restricciones geométricas. En el Capítulo 6 se presenta la paralelización de los dos algoritmos secuenciales para la creación de redes porosas sujetas las restricciones geométricas. En el Capítulo 7 se presentan la plataforma de pruebas y los resultados obtenidos así como una análisis de los mismos; por último, en el Capítulo 8 se presentan las conclusiones y el trabajo futuro.

Capítulo 2

Simulación de Redes Porosas

2.1. Modelado de Redes Porosas

Existen diversos modelos para la simulación de materiales porosos, algunos se enfocan en los procesos o fenómenos que suceden dentro de los mismos y otros en la representación del material por sí mismo, este último tiene la ventaja de que además de poder conocer las características específicas del material también podemos simular procesos y fenómenos dentro del mismo. Uno de los modelos teóricos más utilizados para obtener una adecuada descripción de la estructura y propiedades de un medio poroso es el Modelo Dual de Sitios y Enlaces (MDSE)[1][2]; en este modelo existen dos tipos de huecos o poros: sitios y enlaces, donde cada sitio está conectado con un determinado número de enlaces, C , llamado conectividad de la red y cada enlace permite la conexión entre dos sitios.

Un ejemplo del Modelo Dual de Sitios y Enlaces (MDSE) se puede observar en la Figura 2.1, en la cual se muestra un medio poroso representado por sitios y enlaces interconectados. Existen varias aplicaciones relacionadas con el MDSE algunas de ellas se reportan en [8] y [10]. En particular los algoritmos presentados en [2], [3] y [4] para la creación de redes porosas y en [7] para la simulación de la porosimetría del mercurio se basan en el MDSE.

En este trabajo y como también se utiliza en [2], [3] y [4], definimos una red porosa como una matriz cúbica la cual está formada por un conjunto de elementos; cada elemento contiene un sitio conectado a tres enlaces directos, el sitio también se conecta a tres enlaces de forma indirecta a través de los sitios vecinos, siguiendo una topología tipo toro tridimensional. Por esto se tiene que cada sitio está conectado directa o indirectamente con seis enlaces, es decir la conectividad de la red es de $C = 6$. El tamaño de la red se caracteriza por el parámetro L , el cual representa el número de sitios a lo largo de un borde de la matriz cúbica. Con este modelo se tiene que una red de tamaño L contiene L^3 sitios y $3L^3$ enlaces (como se muestra en la Figura 2.2).

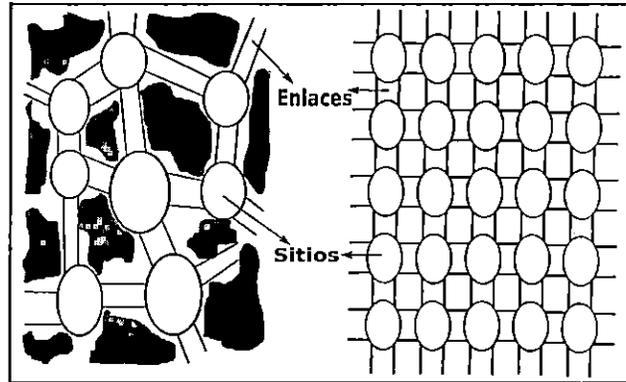


Figura 2.1: Representación bidimensional de un material poroso mediante el MDSE

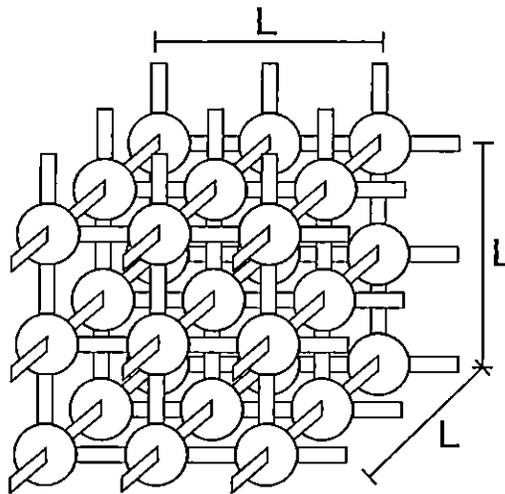


Figura 2.2: Red porosa con los sitios representados por esferas y los enlaces representados por cilindros, con $C = 6$

2.2. Principio de Construcción

Un importante aspecto a considerar en la construcción de redes porosas es el Principio de Construcción (*PC*) en el cual se establece que: El tamaño de cada sitio debe ser mayor o al menos igual al tamaño de cualquiera de los enlaces conectados al mismo. Los tamaños de los poros se representan a través de dos distribuciones normales $F_S(R_S)$ para los sitios y $F_B(R_B)$ para los enlaces. R_S es el radio de la esfera que representa a los sitios y R_B es el radio del cilindro que representa a los enlaces. Dada las anteriores distribuciones se sabe que si $F_S(R_S)$ y $F_B(R_B)$ se traslapan, algunos sitios y enlaces tendrán valores iguales. A esta intersección se le conoce como traslape Ω (Figura 2.3). El traslape representa la dificultad de que los sitios y enlaces se conecten de una forma válida, respetando el *PC*. Si $\Omega = 0$ esto significa que cualquier enlace es menor que cualquier sitio en términos de tamaño, lo que significa que la construcción de una red de poros sería muy sencilla. Por el contrario si Ω es muy cercano a 1, $F_S(R_S)$ y $F_B(R_B)$ serían muy similares por lo que la asignación de enlaces a los sitios estaría más restringida.

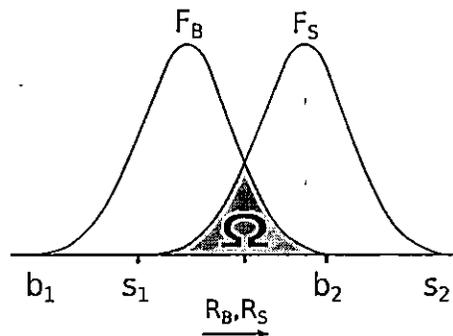


Figura 2.3: Traslape entre las distribuciones F_B y F_S

2.3. Restricciones Geométricas

En la mayor parte de los trabajos relacionados con la creación de redes porosas solo se basan en el *PC* para determinar si una red porosa es válida; sin embargo para obtener redes porosas que se asemejen más a la realidad se deben considerar las Restricciones Geométricas (*RG*) entre sitios y enlaces, lo que complementa al *PC*. Al obtener redes porosas más reales las simulaciones tienen mayor exactitud y los experimentos basados en las mismas tienen mayor probabilidad de éxito.

Para que una red porosa esté sujeta a *RG* se debe cumplir que para cada enlace conectado a un sitio determinado no debe solaparse con otro enlace conectado al mismo sitio, tal y como se muestra en la Figura 2.4. Las restricciones geométricas se aplican cuando se crea una red porosa consistente, mediante el establecimiento de que para cada par de enlaces vecinos conectados a un sitio, la suma de los cuadrados de sus

radios debe ser menor (o a lo más igual) que el cuadrado del sitio. Esto se expresa en la Ecuación 2.1, donde i, j varían desde 1 hasta 6.

$$R_B^2[i] + R_B^2[j] \leq R_S^2 \quad (2.1)$$

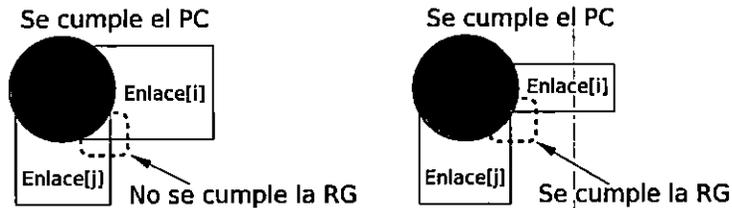


Figura 2.4: Representación del *PC* y *RG* en un sitio con dos enlaces vecinos

La creación de redes porosas de gran tamaño requiere de grandes recursos de cómputo debido a que un medio poroso está compuesto por millones, billones, trillones o más poros por unidad de masa, es por eso que se requieren de soluciones eficientes para poder simular redes porosas de gran tamaño. Debido al volumen de datos que se maneja en las redes porosas las capacidades de cómputo que se necesitan son realmente grandes, lo que se traduce a tiempos de ejecución muy largos. Para dar solución a este problema se pueden aplicar soluciones paralelas para la creación de redes porosas y de esta forma disminuir los tiempos de ejecución y aprovechar las arquitecturas actuales, algunos ejemplos de implementaciones paralelas se pueden ver en [4].

Capítulo 3

Objetivos

El objetivo principal de este trabajo es proponer e implementar un sistema eficiente para la simulación de Redes Porosas sujetas a Restricciones Geométricas. En el presente trabajo se plantea el desarrollo de un sistema paralelo que aproveche las ventajas de las arquitecturas multicore y la programación paralela sobre memoria compartida para acelerar el tiempo de construcción de las redes porosas.

3.1. Objetivos Particulares

- Diseñar e implementar un algoritmo para el particionamiento dinámico de datos en un sistema multi-core.
- Proponer una política de distribución dinámica de datos para permitir que cada tarea computacional pueda trabajar con diferentes datos a lo largo de su ejecución.
- Obtener una comparación de los métodos paralelos propuestos identificando sus ventajas y desventajas.

Capítulo 4

Trabajo Relacionado

Existen varios algoritmos propuestos para la creación de redes porosas que siguen el MDSE y que solo están sujetos al *PC*. Entre los más sobresalientes están el método BiaSED[1] y el algoritmo NoMISS[3]. En este capítulo se presentarán ambos algoritmos junto con sus respectivas versiones paralelas, descritas en [4], las cuales destacan debido a que permiten crear redes porosas de gran tamaño.

4.1. Algoritmos Secuenciales

En esta sección se presentan dos algoritmos secuenciales para la creación de redes porosas sujetas únicamente al *PC*. El primer algoritmo es de tipo iterativo, que hace uso del Método de Monte Carlo [15] y el segundo algoritmo es un algoritmo voraz (o ávido) que construye eficientemente redes porosas.

4.1.1. Algoritmo BiaSED

El algoritmo BiaSED (Biased Simulation Early Design) se describe completamente en [1] y [4]. Se define como un algoritmo iterativo que se compone de tres pasos fundamentales.

Paso 1: Se generan aleatoriamente los tamaños de los sitios (L^3 tamaños de radios de sitios) y los enlaces ($3 * L^3$ tamaños de radios de enlaces), con base en las distribuciones $F_S(R_S)$ y $F_B(R_B)$. Durante la generación, los valores son asignados al azar dentro del espacio cúbico de una red porosa. La asignación se repite hasta que la red esté completamente inicializada. El resultado de este primer paso es una red porosa que probablemente no cumpla con el *PC*, especialmente cuando hay un traslape alto entre $F_S(R_S)$ y $F_B(R_B)$.

Paso 2: Después del paso anterior se tiene una red porosa con $4L^3$ poros (L^3 sitios + $3L^3$ enlaces), sobre la cual se ejecutan una serie de Pasos de Monte Carlo (*MCs*) hasta que se genera una red válida. Un paso de Monte Carlo involucra $4L^3$ intercambios de tamaños de poros, lo anterior para lograr que las conexiones entre los poros sean válidas, es decir, que cumplan con el *PC*. Cada uno de los intercambios se hace cambiando el tamaño de dos sitios o dos enlaces, los cuales se seleccionan aleatoriamente; un intercambio es válido si y solo si el número de violaciones al *PC* es menor o igual al número de violaciones existentes antes del intercambio, de lo contrario el intercambio es rechazado.

Paso 3: Se aplica un número adicional de *MCs* para mejorar la isotropía de la red porosa. El mejoramiento de la isotropía hace que las redes adquieran una distribución de poros más representativa de las redes porosas reales.

4.1.2. Algoritmo NoMISS

NoMISS (No Mistake Initial Seeding Situation) es un algoritmo voraz que trabaja con pequeñas soluciones válidas y que a través de iteraciones llega a una solución válida completa, este algoritmo se describe completamente en [3]. El método se puede separar en cuatro pasos fundamentales:

Paso 1. Generación de Poros: se crean dos listas de sitios L_S y L_{SC} . Los tamaños de radios de los sitios (L^3 radios) son generados de forma aleatoria (con base en la distribución $F_S(R_S)$), los radios se ordenan de forma ascendente y se almacenan en la lista etiquetada como L_S . Después se generan los radios de los enlaces de forma aleatoria (con base en la distribución $F_B(R_B)$). En adelante usaremos el término sitio o enlace para referirnos al radio del sitio o del enlace, respectivamente. Por cada enlace generado, este se intenta conectar al primer sitio de la lista L_S (al más pequeño de la lista), mientras se cumpla el *PC*. Si no es posible, el enlace se intenta conectar con el siguiente sitio en L_S , y así sucesivamente hasta que la conexión cumpla el *PC*. Cada vez que un sitio en L_S completa su contorno, es decir tiene $C = 6$ enlaces conectados, el sitio es trasladado al final de la lista etiquetada como L_{SC} la cual conserva el orden ascendente de los radios de sitios. Al final, L_{SC} contiene sitios con sus respectivos seis enlaces válidos y L_S contiene sitios con conexiones incompletas. Todos los enlaces conectados tanto en los sitios de L_{SC} como en los de L_S siempre cumplen el *PC*.

Paso 2. Sembrado: en este paso se eligen k semillas (es decir k sitios) tomadas de forma aleatoria de la lista L_{SC} , y son insertadas en posiciones aleatorias de la red, entonces los otros sitios de L_{SC} son insertados alrededor de cada semilla, generando así estructuras cúbicas (clusters cúbicos de poros) hasta lograr el tamaño establecido como el cluster-size. En la Figura 4.1 se muestra la construcción de un cluster de tamaño $3 \times 3 \times 3$. Cuando un sitio es conectado a otro sitio en un cluster, sólo se necesita de un enlace para su conexión (el cual debe de cumplir con el

PC). Cuando dos enlaces se encuentran frente a frente para conectar dos sitios y los dos enlaces cumplen con el *PC*, el enlace de mayor tamaño es elegido para la conexión; de otra forma, se elige el enlace más pequeño el cual permite que el *PC* se cumpla en ambos extremos del enlace. El enlace sobrante es asignado a otro sitio en L_S , siguiendo el mismo procedimiento del Paso 1.

Paso 3. Rellenado: una vez terminado el proceso de siembra se comienza el relleno de los espacios vacíos de la red, esto se realiza seleccionando aleatoriamente una de las semillas iniciales y se hace crecer su cluster hasta que se ocupen todos los espacios vacíos de la red. Se hace notar que, si un espacio ya ha sido previamente inicializado, éste es descartado, considerando solamente los espacios vacíos.

Paso 4. Mejoramiento de isotropía: en este punto, igual que en el algoritmo BiaSED, la red porosa ya es válida y solo hay que aplicar un número adicional de *MCs* para mejorar su isotropía.

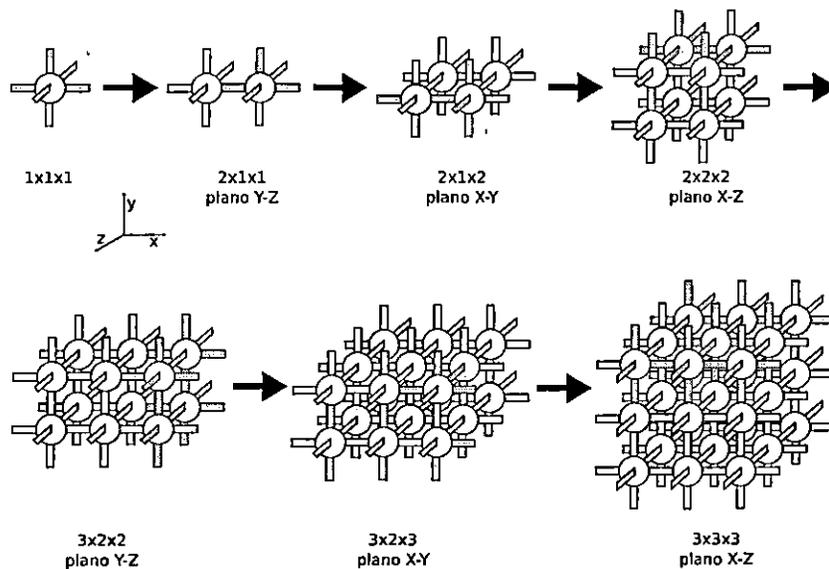


Figura 4.1: Construcción de un cluster de tamaño 3x3x3 con el algoritmo NoMISS

El Algoritmo NoMISS a diferencia del BiaSED, no hace uso de los pasos de Monte Carlo, sino que crea desde el inicio una red porosa válida.

Estos algoritmos que se acaban de presentar ofrecieron buenos resultados al construir redes porosas que respetan el *PC*; sin embargo, ambos algoritmos necesitan tiempos largos para construir redes grandes. Debido a esto, hubo propuestas de paralelización de los dos algoritmos; en la siguiente sección se presenta una descripción breve de las

versiones paralelas de BiaSED y NoMISS, las cuales se encuentran descritas a detalle en [4].

4.2. Algoritmos Paralelos

Las versiones paralelas de BiaSED y NoMISS están diseñadas para trabajar bajo el modelo de paso de mensajes, utilizando la tecnología de Message Passing Interface (MPI-C). En cada versión paralela la red es particionada en pequeñas subredes, como se observa en la Figura 4.2; esta división se hace con base al número de nodos a utilizar, cada nodo tiene una subred de tamaño $L_x \cdot L_y \cdot L_z$, dicha distribución se hace utilizando funciones específicas de MPI. Los nodos mantienen una topología tipo toro, esto para crear una interconexión entre las distintas subredes. Para la explicación de los algoritmos se establece que un nodo es representado por un proceso MPI, por lo que utilizaremos indistintamente ambos términos.

4.2.1. Algoritmo Paralelo BiaSED

El algoritmo BiaSED-paralelo se describe completamente en [4]. Tal y como se comentó al inicio de esta sección, la red porosa se divide en subredes las cuales se distribuyen entre los nodos de un clúster utilizando la tecnología de MPI, como vemos en la Figura 4.2 al utilizar 8 procesos. A continuación se describen a grandes rasgos los pasos del funcionamiento de este algoritmo.

Paso 1: Cada proceso genera L^3/N sitios y $3L^3/N$ enlaces, donde N es el número de procesos a utilizar. Cada proceso ejecuta el paso uno del algoritmo secuencial BiaSED trabajando en su respectivo espacio de subred.

Paso 2: Lo siguiente es que cada proceso ejecuta una serie de *MCs* sobre su subred, a esto se le llama *MCs* paralelo. Este proceso se realiza excluyendo a los sitios ubicados en las caras exteriores de la subred, ya que dichos elementos no cuentan con sus 6 enlaces y no podría validarse el *PC*.

Paso 3: Para que cada poro tenga la posibilidad de intercambiarse con otro de una subred distinta, se realiza una transferencia parcial de subred hacia una dirección específica (x , y o z), entre los nodos del toro que tienen subredes vecinas. De esta manera, los sitios de las caras exteriores quedan ahora en la parte interna de una nueva subred.

Paso 4: Los Pasos 2 y 3 son repetidos, alternando los ejes x , y y z , hasta que se obtiene una red porosa válida.

Paso 5: En este último paso cada nodo repite los Pasos 2 y 3 en su subred un número determinado de veces, para mejorar la isotropía de la red porosa.



Casa abierta al tiempo

UNIVERSIDAD AUTÓNOMA METROPOLITANA
Unidad-Iztapalapa

CBI. AP.1417. 2016
27 de julio de 2016

Dr. Ricardo Marcelín Jiménez
Jefe del Departamento de Ingeniería Eléctrica
UAM - Iztapalapa

P r e s e n t e

De acuerdo con el punto 3 de los Lineamientos Particulares para presentar el examen de grado en los programas de posgrado de la División de C.B.I. - Iztapalapa, que establece la entrega a la jefatura de departamento de un ejemplar de la tesis de Maestría de cada alumno, adjunto a la presente envío a usted la copia de la tesis de Maestría en (Ciencias y Tecnologías de la Información) que presentó el alumno:

Ángel González Méndez

con el título:

“Propuesta de un sistema paralelo para la creación de redes porosas sujetas a restricciones geométricas en arquitecturas multicore”

A t e n t a m e n t e
Casa abierta al tiempo



M. en C. Ma. Iseo González Christen
Asistente de Posgrado

División de Ciencias Básicas e Ingeniería

Edificio Carlos Graeff (T) cubículo 133

4 Av. San Rafael Atlixco No. 186, Col. Vicentina, 09340, México, D.F.

Tel. 5804 4605

Fax: 5612 9162

E-mail: pdiv@xanum.uam.mx

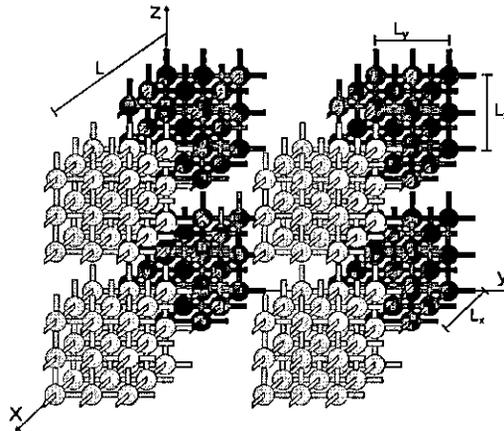


Figura 4.2: Distribución de una red porosa entre 8 nodos o procesos MPI

4.2.2. Algoritmo Paralelo S-NoMISS

El algoritmo paralelo S-NoMISS se describe con mayor detalle en [4], y está basado en el algoritmo NoMISS descrito en la sección anterior. Este algoritmo paralelo se distingue por utilizar un método estático para la distribución de carga, los pasos del algoritmo son los siguientes:

Paso 1. Generación de poros: Cada nodo genera L^3/N sitios y $3L^3/N$ enlaces, donde N es el número de nodos a utilizar. Los sitios se ordenan de forma ascendente. De este modo cada nodo crea las listas locales L_S y L_{SC} , de la misma forma en que se hace en el Paso 1 del algoritmo NoMISS.

Paso 2. Sembrado y Rellenado: Cada nodo ejecuta sobre su subred el paso de sembrado y relleno del algoritmo NoMISS secuencial, con la restricción de que se omite la inicialización en posiciones ubicadas en las caras externas de cada subred.

Paso 3. Rellenado de caras externas: Para rellenar los espacios vacíos existentes entre las subredes y lograr cumplir el PC entre las fronteras de las subredes adyacentes, se realizan un serie de transferencias parciales de subred entre nodos, a lo largo de los ejes x , y y z del toro. Entre cada transferencia se aplica el método de relleno, para que los espacios vacíos que ahora quedaron en la parte interna de una subred se vayan inicializando, respetando el PC . Al terminar las transferencias, se tiene una red porosa válida, sin posiciones vacías.

Paso 4. Mejoramiento de isotropía: En esta última parte se aplica el mismo Paso 5 del método BiaSED-paralelo, para mejorar la isotropía de la red porosa.

Debido a que en este algoritmo los sitios de mayor tamaño quedaban siempre en la parte exterior de la red porosa, era muy difícil mejorar la isotropía de la red, sobre todo cuando se tenían traslapes (Ω) muy altos. Por tal motivo se propuso una nueva versión paralela de NoMISS, D-NoMISS, que permitió que los sitios de mayor tamaño pudieran

distribuirse a lo largo de todo el espacio de la red, como lo hace la versión secuencial NoMISS.

4.2.3. Algoritmo Paralelo D-NoMISS

El algoritmo paralelo D-NoMISS se describe con mayor detalle en [4], este algoritmo ejecuta los pasos del algoritmo S-NoMISS con algunas modificaciones. Los pasos de D-NoMISS son los siguientes:

Paso 1. Generación de poros: Aquí cada proceso ejecuta simultáneamente el mismo paso 1 de S-NoMISS.

Paso 2. Sembrado paralelo: En este paso, el sembrado paralelo de sitios se realiza casi igual que en S-NoMISS, en donde cada proceso omite las caras externas de su subred; lo que cambia ahora es el espacio en donde las semillas son asignadas y agrandadas en clústeres cúbicos de poros. Para que cada poro tenga la posibilidad de ser sembrado en cualquier parte de la red porosa, los procesos transfieren la mitad de su subred a sus respectivos vecinos en el toro (en las tres direcciones x , y y z), y entre cada transferencia se realiza una siembra paralela de sitios. De esta manera los espacios en las caras externas también podrían ser inicializados durante el sembrado. Los clúster de poros ocupan hasta el 25 % de la subred. Cabe destacar que las listas L_{SC} locales de cada nodo, al terminar el sembrado paralelo no son necesariamente del mismo tamaño, debido a los traslapes de clústeres que posiblemente ocurrieron durante el proceso.

Paso 3. Rellenado paralelo: Cada proceso intenta llenar los lugares vacíos de su subred mediante un clúster recubridor que inicia desde el centro de la subred y que se extiende hasta un tamaño $(L_x - 2) \cdot (L_y - 2) \cdot (L_z - 2)$, excluyendo las caras externas de la subred. Debido al paso dos y las posibilidad de que las listas L_{SC} sean de distintos tamaños, cada vez que un proceso se queda sin sitios que asignar, se ejecuta una política de distribución de carga que hace que las listas de poros permanezcan equilibradas (respecto al número de sitios que contienen). Igual que en el Paso 3 de S-NoMISS, para inicializar todos los espacios vacíos se realizan un serie de transferencias parciales de subred entre nodos, a lo largo de los ejes x , y y z del toro, y entre cada transferencia se construye un clúster recubridor.

Paso 4. Mejoramiento de isotropía: Como en todos los algoritmos descritos anteriormente, se recomienda aplicar un número adicional de MCs para mejorar la isotropía de la red porosa; esto se hace de la misma forma que el Paso 4 de S-NoMISS.

4.3. Conclusiones

En general se ha mostrado que los algoritmos NoMISS tienen mejor rendimiento que los algoritmos BiaSED [3], especialmente cuando se pretende construir redes porosas con alto traslape (Ω). En cuanto a las versiones paralelas de NoMISS la versión S-NoMISS se observa mejor en términos de tiempo; sin embargo en términos de la isotropía de la red el algoritmo D-NoMISS es mejor, el mayor problema de la versión D-NoMISS es que su método de distribución dinámica genera un punto de sincronización global que hace que su escalabilidad sea limitada.

El inconveniente de estas versiones es que únicamente toman en cuenta el cumplimiento del Principio de Construcción, por lo que es de gran interés el desarrollo de simuladores de redes porosas que consideren también el cumplimiento de las Restricciones Geométricas (RG) durante la interconexión de poros. En [5] se muestra una primera aproximación para la creación de redes porosas las cuales cumplen completamente con el PC y las RG ; sin embargo, el método propuesto resultó ser una solución impráctica ya que requería de grandes tiempos de ejecución para la generación de redes porosas de tamaños de red relativamente pequeños (que contenían a lo más 40^3 poros). Otra aproximación es la que proponen en [17] donde describen un algoritmo secuencial híbrido que inicialmente ejecuta un procedimiento tipo voraz para inicializar la red y posteriormente aplica un método iterativo para eliminar las violaciones a las restricciones de PC y RG . La desventaja de este algoritmo son sus tiempos de ejecución para crear redes de gran tamaño debido a procesos de ordenamiento y de búsqueda que tuvieron que incluirse.

En búsqueda de algoritmos eficientes que consideren el cumplimiento de ambas restricciones, PC y RG , como trabajo preliminar se probó la adaptación del algoritmo secuencial de NoMISS (NoMISS- RG) para crear redes de porosas que cumplan con los dos tipos de restricciones. Sin embargo, debido a que los enlaces y sitios se generan de forma aleatoria fue muy difícil encontrar configuraciones válidas, lo que se transformaba en tiempos de ejecución muy altos y como resultado solo se logró generar redes porosas con traslapes muy pequeños ($\Omega \leq 0,0007$) utilizando este esquema. Con base en el resultado anterior no se intentó paralelizar dicha versión ya que la limitante del traslape afectaría de igual forma a una versión paralela.

Derivado de las limitaciones en términos de escalabilidad de [5], [17] y NoMISS- RG para la creación de redes porosas que cumplan tanto con el PC y las RG . En este trabajo se proponen dos versiones paralelas para la construcción de redes porosas que consideran el cumplimiento de PC y RG los cuales están inspiradas en los algoritmos BiaSED y NoMISS. En la primera versión se propone paralelizar el algoritmo BiaSED incluyendo el cumplimiento de las RG . En la segunda versión se propone paralelizar el algoritmo descrito en [17] en donde el procesamiento paralelo se complementa con una política de distribución dinámica de datos la cual permita que cada tarea computacional (hilo) pueda trabajar prácticamente en cualquier zona de la red lo que da como resultado una red porosa con una mejor isotropía. Ambas propuestas fueron

desarrolladas utilizando arquitecturas multi-núcleo, para sacar el mayor provecho de la memoria compartida entre procesadores. Cabe destacar que si bien nos hemos inspirado en los algoritmos BiaSED y NoMISS no se deben de tomar en cuenta como punto de partida o comparación ya que el trabajo propuesto implementa un algoritmo para el cumplimiento de las *RG* lo que se transforma en un mayor procesamiento el cual no es tomado en cuenta por las versiones secuenciales o paralelas de BiaSED y NoMISS.

Capítulo 5

Construcción Secuencial de Redes Porosas sujetas a RG

En este capítulo se describen dos algoritmos secuenciales para la creación de redes porosas sujetas a restricciones geométricas las cuales se definieron en la Sección 2.3, en donde el valor de traslape (Ω) alcanzado por las redes que cumplan tanto con el PC y las RG será relativamente bajo ($\Omega < 1$) este hecho se debe a la restricción impuesta por las mismas restricciones geométricas lo cual se ilustra en las ecuaciones 5.1 y 5.2.

$$B_C(R_S) \geq S(R_S) \quad (5.1)$$

$$B_C(R_S) = \left\{ \int_0^{R_S} \dots \int_0^{R_S} F_B(R_{B1}) \dots F_B(R_{BC}) dR_{B1} \dots dR_{BC} \right\}^{1/C} \quad (5.2)$$

donde $B_C(R_S)$ corresponde al volumen definido por la ecuación 5.2; esta se relaciona con el conjunto de enlaces que pueden ser conectados a un sitio, evitando al mismo tiempo la existencia de interferencias entre ellos. $S(R_S)$ es la fracción de sitios que son de tamaño R_S o más pequeños.

Las ecuaciones 5.1 y 5.2 restringen el valor de traslape como se muestra en [5], ya que la definición matemática de $B_C(R_S)$ impide en si alcanzar valores cercanos a la unidad.

En ambos algoritmos la generación de sitios y enlaces se basa en las distribuciones $F_S(R_S)$ y $F_B(R_B)$. Ambos algoritmos propuestos tienen tres grandes fases principales: inicialización de la red porosa con sitios y enlaces, la generación de una red porosa válida y el mejoramiento de la isotropía. La diferencia principal entre ambos algoritmos radica en el fase de inicialización, en donde la primera solución llamada Solución

Aleatoria(*SA*) inserta de forma aleatoria sitios y enlaces hasta inicializar toda la red porosa mientras que la segunda solución llamada Solución Híbrida(*SH*) inserta grupos de sitios (clusters) a lo largo de la red y posterior mente se les asignan enlaces a los sitios intentando siempre cumplir con el *PC* y las *RG*.

Después de la fase de inicialización, es posible que ambos algoritmos dieran como resultado redes porosas que tienen violaciones a las *RG* el número de violaciones a las *RG* dependerá directamente del valor del traslape (Ω), por lo que la segunda fase consiste en la eliminación de dichas violaciones utilizando el Método de Monte Carlo. Por último, la tercera fase se encarga del mejoramiento de la isotropía de las redes porosas haciendo uso de igual manera del Método de Monte Carlo. A continuación se describe a detalle cada algoritmo.

5.1. Solución Aleatoria

En esta sección se describe el algoritmo de inicialización para la Solución Aleatoria(*SA*), este algoritmo tiene por objetivo inicializar la red porosa con sitios y enlaces de forma aleatoria en dos pasos, en el primer paso se generan e insertan en la red porosa L^3 tamaños de radio de sitios, con base en la distribución $F_S(R_S)$; conforme se van generando los sitios éstos se insertan en la red porosa. El segundo paso consiste en conectar tres enlaces a cada sitio de la red porosa; cada enlace es generado con base en la distribución $F_B(R_B)$ y al final tendremos $3 \cdot L^3$ enlaces dentro de la red porosa. Al finalizar esta fase se obtendrá una red porosa que puede tener violaciones a las *RG*, el número de violaciones depende directamente del valor del traslape(Ω).

Cabe destacar que aun cuando se le asignaron solo tres enlaces a cada sitio, la conectividad de estos sigue siendo igual a seis debido a que los otros tres enlaces los comparte con sus sitios vecinos siguiendo una topología tipo toro, tal y como se muestra en la Figura 5.1. Se puede apreciar que al sitio de color gris oscuro le comparten un enlace cada uno de sus sitios vecinos que están representados en color gris claro, a su vez el sitio de color gris oscuro le comparte sus enlaces a los sitios que se encuentran en la posición frente-inferior-derecho, trasero-superior-derecho y frente-superior-izquierdo de la red porosa.

5.2. Solución Híbrida

En la presente sección se describe el algoritmo de inicialización para la Solución Híbrida(*SH*), el algoritmo que se presenta a continuación inicializa una red porosa con sitios y enlaces donde se intenta que la conexión entre estos cumpla con las *RG*. La primera acción del algoritmo es crear dos listas de tamaños de poros ordenadas de forma descendente (con base al tamaño de los sitios y enlaces, respectivamente) mediante un algoritmo de ordenamiento por inserción. De esta manera, el primer sitio/enlace de

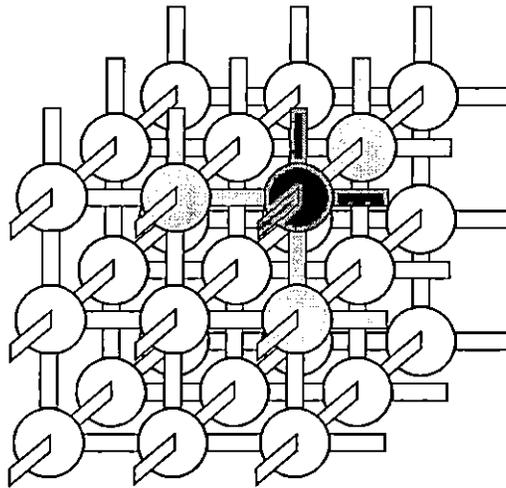


Figura 5.1: Red porosa inicializada con L^3 sitios y $3 \cdot L^3$ enlaces, con $L=3$.

cada lista es el que tiene el mayor tamaño respecto a los otros elementos subsecuentes.

La primera lista L_S se compone de L^3 radios de sitios, mientras que la segunda, L_B , contiene $3L^3$ enlaces. Después de la creación de las listas se lleva a cabo el proceso de sembrado y relleno de sitios similar al procedimiento que realiza el algoritmo NoMISS descrito en la Sección 4.1.2. Al finalizar esta parte, sigue el proceso de asignación de enlaces a los sitios. A continuación se describe a detalle el algoritmo.

5.2.1. Sembrado de Clusters de sitios

Durante todo el proceso de sembrado y relleno de sitios en la red porosa cada vez que se toma un elemento de la lista L_S se hace referencia a que se toma el primer elemento actual de L_S , siendo este el sitio actual más grande.

El proceso de sembrado de clusters se divide en dos pasos, el primero consiste en tomar el primer sitio (semilla) de la lista L_S e insertarlo en una posición aleatoria de la red porosa. El segundo paso consiste en tomar más elementos de L_S y uno a uno insertarlos alrededor de la semilla actual hasta crear un cluster de sitios de tamaño $ClusterSize$, el cual tendrá la forma de un cubo; este procedimiento se describe en las líneas 1-6 del Algoritmo 1. En la Figura 5.2 se muestra de forma gráfica la creación de un cluster donde $ClusterSize = 3$.

Cada vez que se inserta un elemento de L_S en una posición aleatoria (i, j, k) de la red, dicha posición se almacena en la lista L_{SC} que se mantiene ordenada de forma descendente con base al tamaño de los sitios insertados. El procedimiento de inserción de semillas y la creación de un cluster de tamaño $ClusterSize$ alrededor de cada semilla se repite p veces, donde p es el número de clusters a insertar. En caso que durante la creación de los clusters exista un traslape entre ellos, el sembrado se omite en las posiciones ya ocupadas (en las que existe traslape) y se continua el sembrado de los sitios en los espacios aun vacíos.

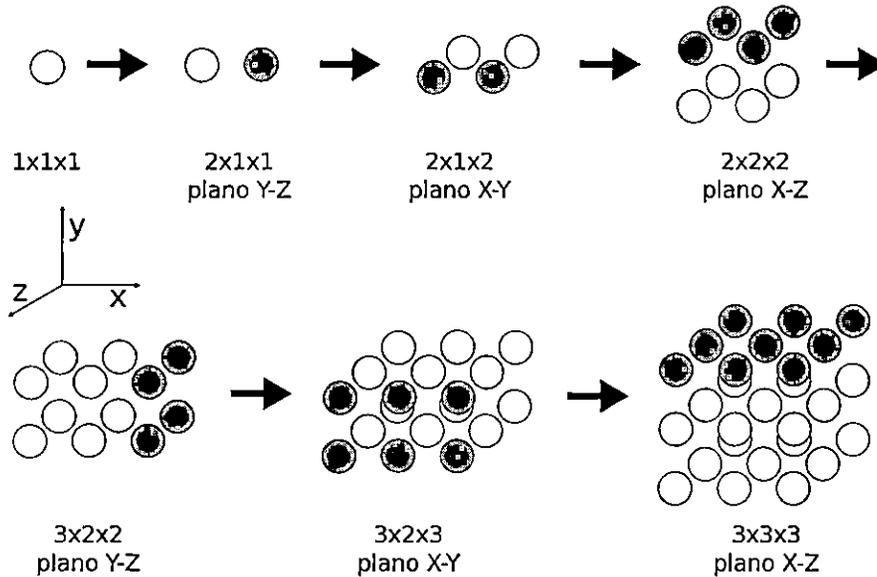


Figura 5.2: Sembrado de un cluster de sitios de tamaño 3x3x3

En la Figura 5.3 se representa una red porosa después del proceso de siembra de p clusters de tamaño 3x3x3 cada uno. También se puede observar que en la red quedan espacios vacíos los cuales serán rellenados de la siguiente forma: una vez completado el proceso de siembra de clusters, al primer cluster generado se le insertan alrededor suyo todos los sitios restantes de L_S (líneas 7-9 del Algoritmo 1) siguiendo las mismas reglas de construcción de cluster anteriores, con la diferencia de que se establece $ClusterSize = L$. Con esto se garantiza que todos los espacios vacíos de la red porosa serán inicializados.

Algoritmo 1 Sembrado de clusters dentro de la red porosa($pnet$)

Require: L_S , L_{SC} , $NClusters$, $ClusterSize$

- 1: for $m \leftarrow 1$ to $NClusters$ do
 - 2: $pnet[i][j][k] \leftarrow FIRST(L_S)$ // i, j, k : posición aleatoria
 - 3: for $p \leftarrow 1$ to $ClusterSize$ do
 - 4: INSERTPCLUSTER($pnet, i, j, k, \&L_S, \&L_{SC}$)
 - 5: end for
 - 6: end for
 - 7: while $SIZE(L_S) > 0$ do // i, j, k : La posición de la primera semilla
 - 8: INSERTPCLUSTER($pnet, i, j, k, \&L_S, \&L_{SC}$)
 - 9: end while
-

Al final del proceso de siembra y relleno, los sitios en su mayoría tienen a su alrededor sitios con tamaños similares, con lo cual se tiene mayor oportunidad de encontrar soluciones válidas al conectar los sitios con los enlaces y que cumplan con las

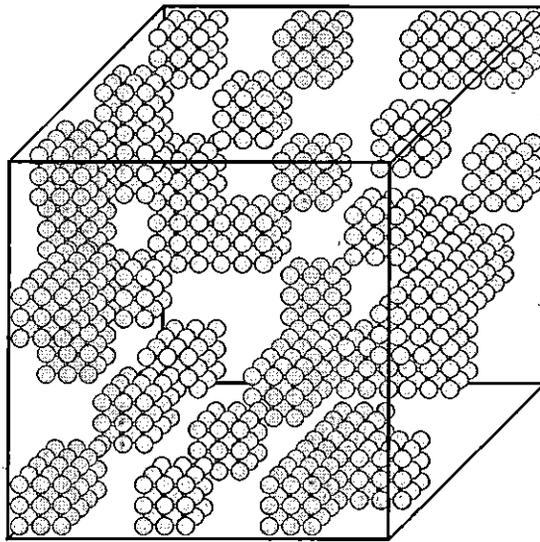


Figura 5.3: Red porosa después del proceso de siembra de clusters

RG. Esto se debe a que es más sencillo encontrar un enlace para conectar dos sitios de tamaño similar a encontrar un enlace que conecte dos sitios de tamaños distintos (o muy distintos). En contraste con las versiones de NoMISS descritas en el Capítulo 4, en este algoritmo los sitios insertados durante el sembrado y rellenado no tienen aún enlaces asignados.

5.2.2. Asignación de Enlaces

La asignación de enlaces consiste en asignar 6 enlaces, primero a los sitios más grandes, luego se continúa con los sitios más pequeños de la red. Para este fin, los elementos de la lista L_{SC} son tomados uno a uno desde la primera posición de la lista. Como se mencionó anteriormente, cada elemento de L_{SC} contiene la posición en la cual cada sitio fue asignado dentro de la red porosa, y el orden de los elementos es descendente con base al tamaño de los sitios. Entonces el primer elemento de la lista siempre almacenará la posición dentro de la red del sitio más grande de la lista. Cuando una posición es tomada de L_{SC} , se le intenta conectar $C = 6$ enlaces al sitio en dicha posición; los enlaces son tomados de la lista L_B , intentando usar los enlaces de mayor tamaño primero. Cada uno de los C enlaces conectados a los sitios debería cumplir con el *PC* y las *RG*. Si el primer enlace de L_B no permite que se cumplan las restricciones respecto a los enlaces previamente asignados al sitio, se intenta tomar al siguiente enlace en la lista, así hasta encontrar un enlace que permita el cumplimiento de las restricciones *PC* y *RG*. Solo en el caso que no exista un enlace en L_B que cumpla con estas restricciones, para completar el contorno de enlaces se toma el enlace más grande de L_B para ser conectado con el sitio actual. Lo anterior ocasiona la existencia de violaciones a las *RG* en la inicialización de la red. En el Algoritmo 2 se muestra este comportamiento.

Algoritmo 2 Asignación de enlaces

Require: $L_B, L_{SC}, C = 6, pnet$

- 1: while SIZE(L_{SC}) > 0 do
 - 2: $(i, j, k) \leftarrow \text{FIRST}(L_{SC})$ // se retira el primer elemento de la lista
 - 3: for $p \leftarrow 1$ to C do // $C = 6$ para una red cúbica
 - 4: ASSIGNBONDRG($pnet, i, j, k, p, \&L_B$)
 - 5: end for
 - 6: end while
-

5.3. Generación de una Red Porosa válida

Una vez que la red porosa ha sido inicializada por completo, ya sea con el algoritmo de la Solución Básica Aleatoria o con el de la Solución Híbrida, la red porosa puede tener violaciones a las RG . Como se explicó anteriormente, la Solución Básica Aleatoria no verifica el cumplimiento de las RG al asignar un enlace a un sitio debido a que es un método completamente aleatorio. Por otra parte, la Solución Híbrida intenta conectar los enlaces grandes con los sitios mas grandes posibles, donde la conexión entre sitio y enlace cumplan con las RG ; sin embargo, existen casos en los cuales no es posible que se cumplan las RG y por consiguiente la conexión provoca violaciones a las RG . Cabe destacar que para ambos algoritmos, el número de violaciones a las RG está directamente relacionado con el valor del traslape.

Para eliminar las violaciones al PC y RG los algoritmos aplican un número sucesivo de pasos de Monte Carlo, hasta obtener una red válida, tal y como lo hace el algoritmo secuencial BiaSED descrito en la Sección 4.1.1. A diferencia del algoritmo BiaSED, los algoritmos previamente descritos en las Secciones 5.1 y 5.2 también consideran el cumplimiento de RG en cada uno de los intercambios de poros, si esto no ocurre, el intercambio es rechazado. La fase para la generación de una red porosa válida termina cuando todos los poros en la red cumplen tanto con el PC como con las RG , como se muestra en el Algoritmo 3.

Algoritmo 3 Esquema de genración de una red porosa válida

Require: $pnet$

- 1: while GRVIOLATIONS($pnet$) > 0 do
 - 2: POREXCHANGE($pnet$)
 - 3: if NONVALIDEXCHANGE($pnet$) then
 - 4: REJECTEXCHANGE($pnet$)
 - 5: end if
 - 6: end while
-

En las Figuras 5.4 y 5.5 se muestra un ejemplo del intercambio válido entre dos sitios y entre dos enlaces respectivamente. En las Figuras 5.6 y 5.7 se muestra un ejemplo de intercambio inválido entre dos sitios y entre dos enlaces, respectivamente. El tiempo

que lleva el proceso de eliminación de violaciones al PC y RG varia dependiendo del traslape entre las distribuciones utilizadas para generar a los tamaños de sitios como de los enlaces de la red porosa, y si este mismo es muy grande puede que el Algoritmo 3 no termine.

5.4. Mejoramiento de la isotropía

Para que una red porosa válida sea lo más cercana a la realidad se requiere tanto del cumplimiento de las restricciones geométricas así como tener una buena isotropía, es decir que los distintos tamaños de los poros estén lo mejor distribuidos en la red. Para esto, después de generar una red porosa válida se aplican una número extra de pasos de MC , siguiendo las mismas reglas utilizadas en la sección anterior. Cabe destacar que el número extra de pasos de MC necesarios para mejorar la isotropía hasta ahora ha sido determinado de manera experimental y depende directamente del traslape(Ω) entre sitios y enlaces.

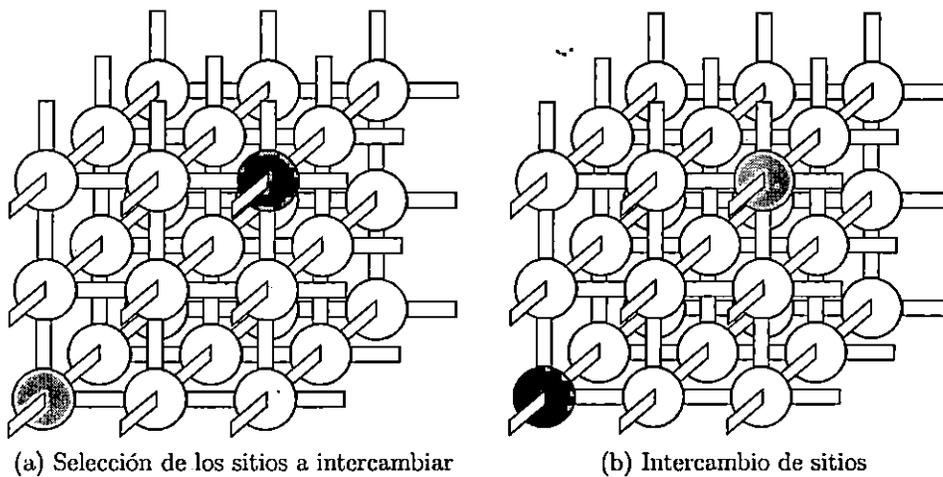


Figura 5.4: Ejemplo de un intercambio válido de dos sitios (a) selección y (b) intercambio

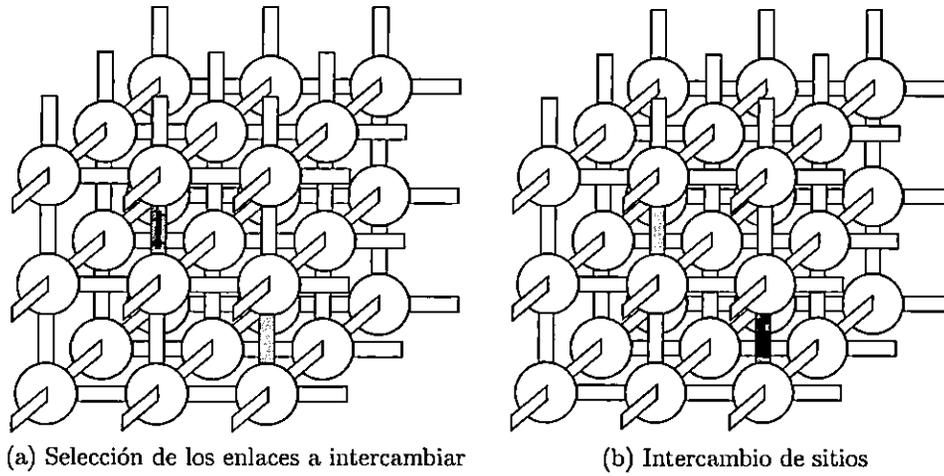


Figura 5.5: Ejemplo de un intercambio válido de dos enlaces (a) selección y (b) intercambio

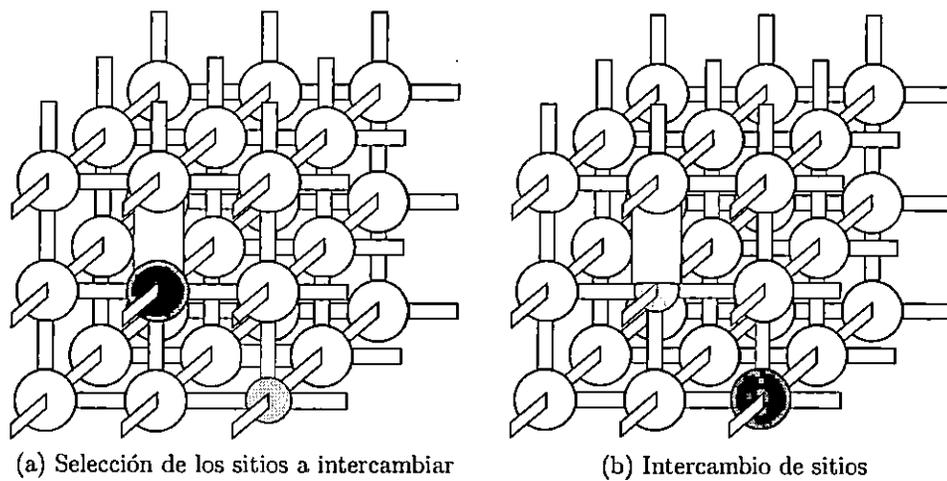


Figura 5.6: Ejemplo de un intercambio inválido de dos sitios (a) selección y (b) intercambio

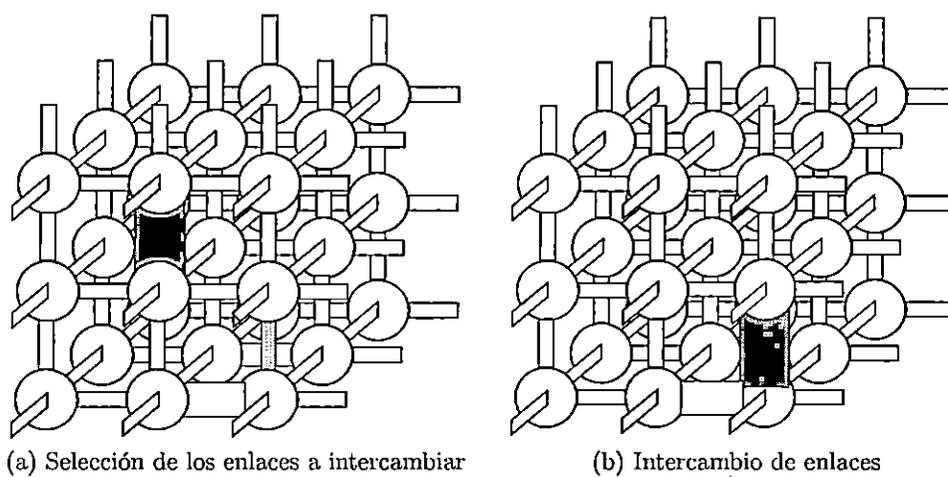


Figura 5.7: Ejemplo de un intercambio inválido de dos enlaces (a) selección y (b) intercambio

Capítulo 6

Construcción Paralela de Redes Porosas sujetas a RG

En este capítulo se describe la propuesta paralela de los dos algoritmos descritos en el capítulo 5 para la creación de redes porosas sujetas a RG . Las propuestas paralelas trabajan sobre el modelo de memoria compartida, donde varios hilos cooperan en la construcción de una red porosa válida siguiendo los principales pasos de los algoritmos presentados en el capítulo anterior. El primer paso para las dos soluciones paralelas es la distribución espacial de la red entre los hilos a utilizar (dicha distribución es igual para ambos algoritmos). Después de la distribución espacial de red cada algoritmo comienza con su respectivo procedimiento de construcción (inicialización, eliminación de violaciones y mejoramiento de la isotropía) en paralelo. Primero presentamos el algoritmo común de distribución espacial y luego la paralelización de las etapas de construcción.

6.1. Distribución Dinámica de la Red Porosa

En este paso una red porosa es dividida en N subredes donde N es el número de hilos a utilizar. El objetivo es dividir la red en N partes las cuales deben mantener una estructura lo más parecida a un cubo. Para lograr esto la red se divide a lo largo de los tres ejes x, y y z en a, b , y c partes, respectivamente. Se debe cumplir que el producto de a, b , y c es igual a N . El tamaño de cada subred esta definido por $L_x \cdot L_y \cdot L_z$, como se puede observar en la Figura 6.1. Por ejemplo, si tomamos $N = 4$, una configuración generada sería $a = 1, b = 2, c = 2$, mientras que para $N = 45$ una configuración generada sería $a = 3, b = 3, c = 5$, y para $N = 27$ una configuración generada sería $a = 3, b = 3, c = 3$, que representa la raíz cubica de 27. La mejor distribución se da cuando a, b , y c corresponden a la raíz cubica de N ; de esta forma las subredes generadas adoptan una estructura cúbica que ayuda a que la distribución de los poros sea equitativa a lo largo de los ejes x, y y z de cada subred. En la Figura 6.2 se muestran algunos ejemplos de particionamiento.

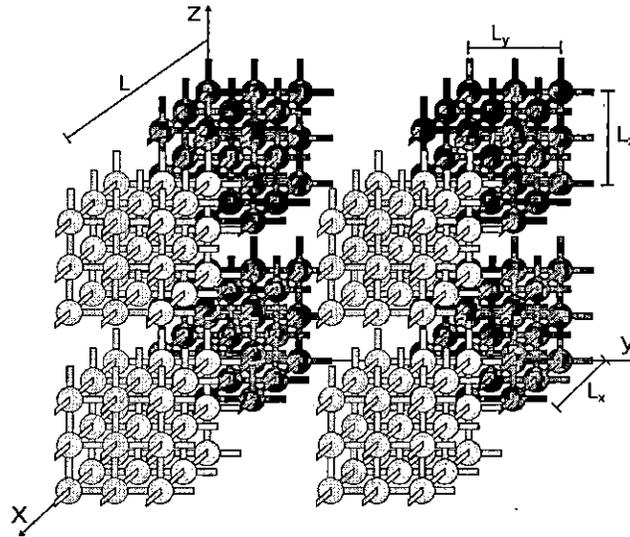


Figura 6.1: Red porosa dividida en ocho subredes; $N = 2 \cdot 2 \cdot 2$

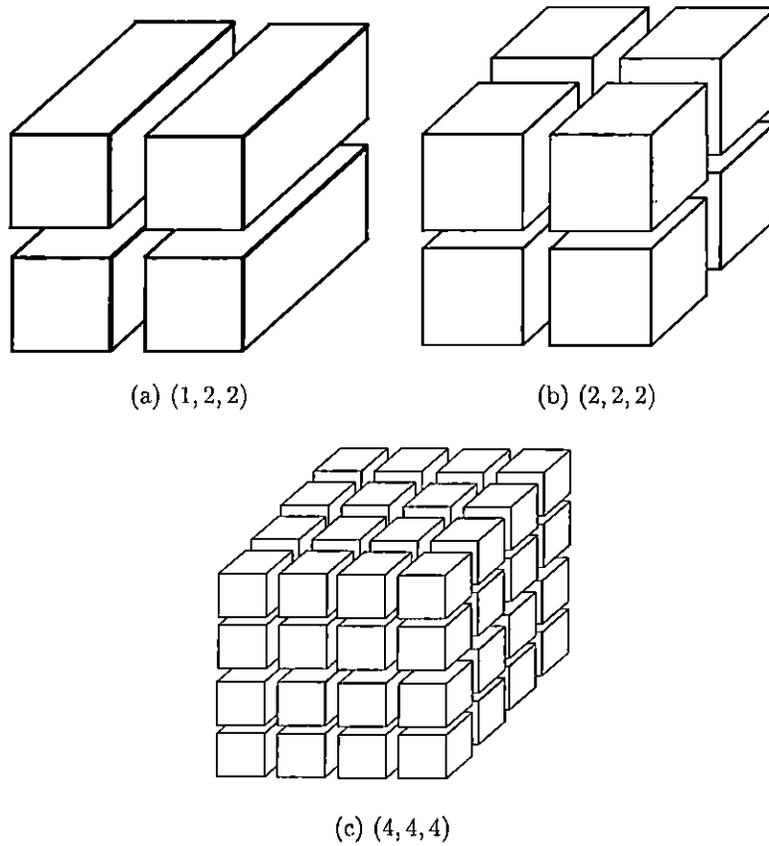


Figura 6.2: Distribución de la Red para (a) 4 hilos, (b) 8 hilos, y (c) 64 hilos

Para permitir que la distribución del espacio de la red porosa no sea siempre el mismo, en este trabajo se propone una distribución dinámica en donde el particionamiento no siempre empieza a partir del origen $(0,0,0)$ de la red. Se propone un cambio de origen con el propósito de obtener un desplazamiento lógico del espacio de la red, para que los hilos puedan trabajar en subredes independientes y diferentes de la red porosa. En la Figura 6.3 se muestra un red porosa distribuida a partir del origen $(0,0,0)$ entre dos hilos (Figura 6.3a) y un ejemplo de un desplazamiento lógico sobre y , con cambio de origen a $(0,2,0)$, puede verse en la Figura 6.3a. En esa figura se observa que las subredes sobre las cuales trabajarán los hilos han cambiado, esta es una de las grandes ventajas de trabajar sobre memoria compartida ya que el tiempo que toma un desplazamiento es totalmente despreciable ya que en ningún momento existe algún tipo de transferencia física de los datos. Al contrario, si se utilizase memoria distribuida esta operación de desplazamiento se traduciría en transferencia de datos entre nodos, tal y como se comentó en la Sección 4.2, dicha transferencia podría transformarse en un cuello de botella.

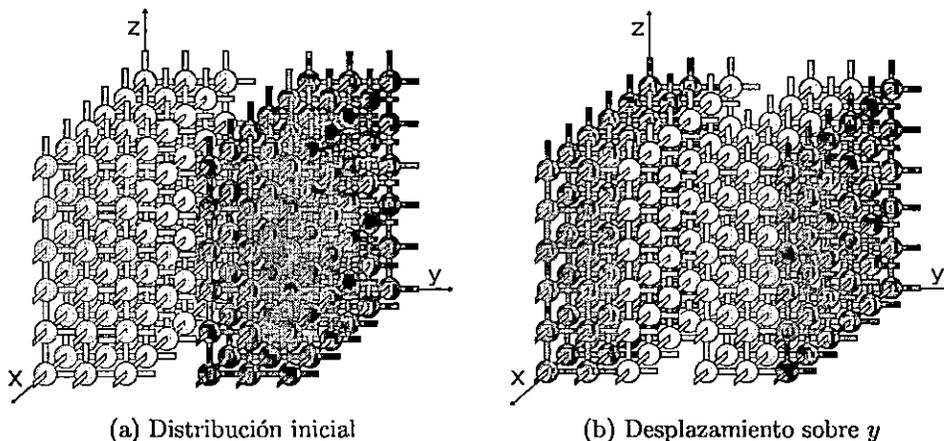


Figura 6.3: Ejemplo de un desplazamiento lógico (a) Posición inicial $(0,0,0)$, (b) desplazamiento sobre y $(0,2,0)$

6.2. Solución Paralela Aleatoria

En esta sección se describe el algoritmo paralelo Solución Paralela Aleatoria (SPA) el cual corresponde a su respectiva versión secuencial descrita en la Sección 5.1.

6.2.1. Inicialización

Una vez que se determina la distribución del espacio de la red porosa entre N hilos como se presenta en la sección anterior, cada hilo trabaja de forma independiente en una de las subredes de tamaño $L_x \cdot L_y \cdot L_z$; la Figura 6.1 muestra un ejemplo de distribución de espacio poroso entre 8 hilos. En el proceso de inicialización cada hilo inserta de

forma aleatoria en su subred $L_x \cdot L_y \cdot L_z$ sitios hasta rellenar por completo la red. Una vez que la red esta inicializada con sitios, a cada sitio se le asignan únicamente tres enlaces: izquierdo, trasero y superior. La asignación de dichos enlaces permite inicializar el contorno completo de cada sitio, evitando la sincronización entre los hilos que accesan las caras externas de subredes adyacentes dentro de la red porosa global.

6.2.2. Generación de una Red Porosa Válida

Después de que la red porosa ha sido inicializada ésta puede contener violaciones a las *RG* ya que al igual que su respectiva versión secuencial la inicialización no verifica el cumplimiento de las *RG* al asignar un valor a los enlaces. En seguida, para eliminar las violaciones se aplica una serie de *MCs* Paralelos; este comportamiento se resume en el Algoritmo 4. Cada paso de *MC* Paralelo implica que cada hilo aplicará un paso de *MC* normal tal y como se hace en el algoritmo secuencial descrito en la Sección 5.3, trabajando sobre su respectiva subred y omitiendo las caras externas (lineas 7-12 del Algoritmo 4). Al omitir las caras externas se evita la sincronización entre hilos que poseen subredes adyacentes.

Para que cada poro(sitio o enlace) de las caras externas de cada subred tenga la posibilidad de intercambiarse con otro, se realizan desplazamientos lógicos a lo largo de los ejes x , y y z después de ejecutar cada paso de *MC* Paralelo (línea 17 del algoritmo 4). Con dichos desplazamientos se logra que caras externas en un momento queden al interior de una subred. Al igual que en la versión secuencial los pasos de *MC* Paralelos se repetirán hasta eliminar por completo las violaciones a las *RG*.

6.2.3. Mejoramiento de la Isotropía

Concluido en procedimiento de la sección anterior se tiene una red porosa válida sin violaciones a la *PC* y a las *RG*, a dicha red se le aplican una serie adicional (K) de *MCs* Paralelos para mejorar la isotropía de la red.

6.3. Solución Paralela Híbrida

A continuación describiremos el algoritmo paralelo Solución Paralela Híbrida (*SPH*) correspondiente al algoritmo secuencial descrito en la Sección 5.2. Tal y como su versión secuencial el algoritmo presentado en esta sección se compone de 3 principales fases: inicialización (sembrado de clusters de sitios y asignación de enlaces), generación de una red porosa válida y mejoramiento de isotropía. El algoritmo hace uso del algoritmo de distribución dinámica descrito en la Sección 6.1 para distribuir la red porosa entre los N hilos a utilizar. Cada hilo ejecuta en paralelo la etapa de inicialización, luego se aplican sucesivos pasos de *MC* Paralelos Modificados para eliminar las violaciones a las *RG*. Al finalizar el algoritmo obtiene una red porosa libre de violaciones a las *RG*.

Algoritmo 4 Algoritmo paralelo de *MC* para generar una red porosa válida

Require: *pnet*: red porosa

Require: *dist*: Arreglo con la distribución de la red

```

1: continue ← GRVIOLATIONS(&pnet) > 0
2: omp parallel private (tid, nMCs, exchanges)
3:   tid ← OMPGETTID( )
4:   nMCs = 4((dist[tid].Lx - 2) · (dist[tid].Ly - 2) · (dist[tid].Lz - 2))3
5:   while continue do
6:     exchanges ← nMCs
7:     while (exchanges --) > 0 do
8:       POREXCHANGE(&pnet, &dist[tid])
9:       if NONVALIDEXCHANGE(&pnet, &dist[tid]) then
10:        REJECTEXCHANGE(pnet, dist[tid])
11:      end if
12:    end while
13:    omp barrier
14:    omp master
15:      continue ← GRVIOLATIONS(&pnet) > 0
16:      if continue then
17:        SHIFT(&pnet, &dist[tid]) // desplazamiento sobre (x, y, z)
18:      end if
19:    end omp master
20:    omp barrier
21:  end while
22: end omp parallel

```

6.3.1. Inicialización

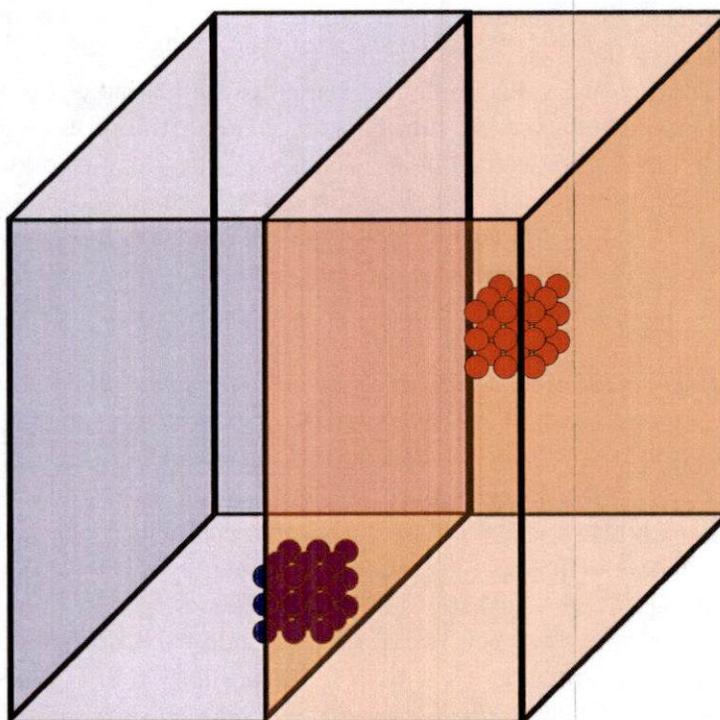
Con base en la distribución de la red cada hilo trabaja de forma independiente en una de las subredes de tamaño $L_x \cdot L_y \cdot L_z$, en esta versión Híbrida los hilos trabajan en subredes diferentes. En la inicialización cada hilo genera dos listas ordenadas L_S y L_B , también se crea la lista L_{SC} la cual inicialmente estará vacía. L_S contiene L' elementos de tamaños de sitios ordenados de forma descendente, y L_B contiene $3L'$ elementos de tamaños de enlaces ordenados también de forma descendente, donde L' es igual a $L_x \cdot L_y \cdot L_z$. Cabe destacar que L' es local a cada hilo ya que aun cuando el algoritmo de distribución de la red intenta que las subredes sean del mismo tamaño, existen casos en los cuales no es posible, ésto da como resultado subredes de diferentes tamaños. La sincronización entre los hilos no es necesaria para el uso o modificación de las listas L_S , L_{SC} y L_B porque estas listas son locales a cada hilo.

Sembrado de Clusters de sitios

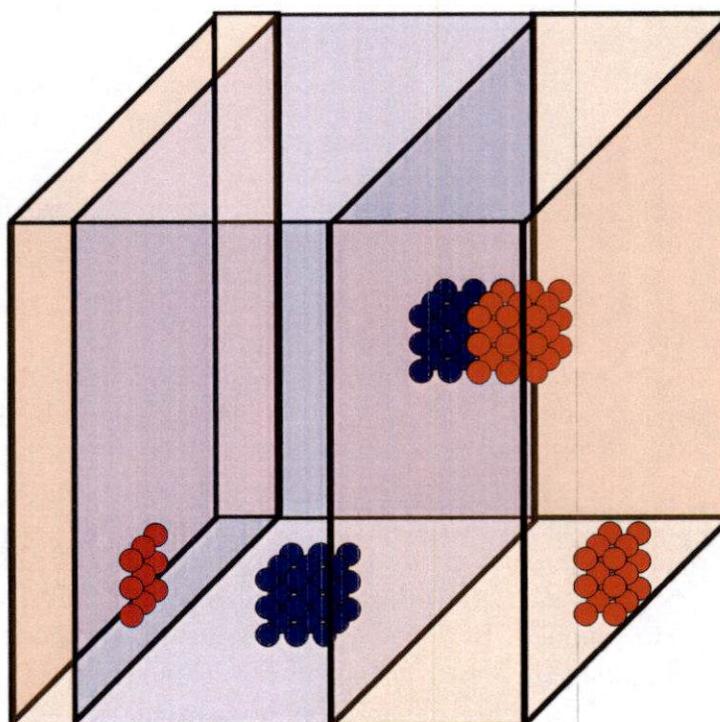
Por cada hilo, el número de semillas corresponde a $NClusters/N$. Cada semilla es insertada en una posición aleatoria en su respectiva subred. Los demás elementos en L_S son tomados uno por uno para cubrir la semilla y formar un cluster, tal y como se explica en el capítulo anterior. La posición aleatoria en la cual se asigna la semilla en la subred es generada de tal forma que no puede ser asignada en las caras externas de la subred. Durante el sembrado todos los hilos se sincronizan a través de una *barrera* después de que cada uno complete un cluster, entonces el origen (x, y, z) de distribución se desplaza de forma aleatoria a lo largo de los ejes x, y y z en un rango que esta entre 0 y $L - 1$. El cambio de origen de distribución nos ayuda a que cada hilo tenga la posibilidad de sembrar un cluster en cualquier posición de la red porosa.

El la Figura 6.4 se muestra un ejemplo del cambio de origen durante el proceso de sembrado. La Figura 6.4a muestra dos clusters de sitios construidos al tener el origen de la red porosa inicialmente en $(0, 0, 0)$. Si movemos el origen a $(0, L/2, 0)$, esto causa que las áreas de trabajo(subredes) de cada hilo cambien para construir en paralelo otro cluster de sitios, como se muestra en la Figura 6.4b. Las áreas de trabajo de cada hilo se resaltan en color azul y naranja, antes y después del desplazamiento. El cambio de origen, además de permitir que las caras externas de cada subred sean tomadas en cuenta durante el sembrado, nos permite hacer parecer(emular) que la red porosa fue inicializada por un solo hilo como en la versión secuencial, ya que prácticamente cada hilo podría trabajar en algún momento en cualquier parte de la red porosa.

Después de completar el sembrado de clusters, el origen de la red es trasladado a $(0, 0, 0)$ de manera que cada hilo comienza a trabajar en su subred inicial. El siguiente paso consiste en rellenar su actual subred con los sitios restantes en su lista L_S ; sin embargo, debido a posibles traslapes durante el sembrado de clusters, en este momento el número de sitios en las listas L_S de cada hilo es diferente. Puede ser que para algunos hilos los sitios que tienen en sus listas L_S no sean suficientes para rellenar su subred, mientras que para otros el número de sitios en sus listas L_S puede ser mayor al de los requeridos para rellenar su subred. Para solucionar este problema los elementos de las



(a) Distribución inicial del espacio



(b) Distribución posterior del espacio

Figura 6.4: Cambio de origen durante el sembrado, (a) Origen en $(0, 0, 0)$ y (b) cambio de origen a $(0, L/2, 0)$

listas L_S requieren ser distribuidos entre los hilos.

Los pasos necesarios para distribuir los elementos de las listas L_S se presentan en el Algoritmo 5 el cual es una versión simplificada del algoritmo presentado en [11] para la distribución de carga. Por cada hilo se tiene, identificador del hilo (id), el número $M[id]$ de sitios requeridos el cual depende de: el tamaño de la subred $SNS[id]$ ($SNS[id] = L_x \cdot L_y \cdot L_z$), el número de sitios que ya han sido insertados en la subred $SI[id]$, y el número de sitios en su lista $AL_S[id]$, denotado por $Size(L_S[id])$. En el algoritmo el arreglo M tiene contexto global por lo cual es una variable compartida.

La primera etapa consiste en el cálculo del número de sitios requeridos; cada hilo calcula $M[id]$ como se indica en la línea 1 del Algoritmo 5. En el segundo paso, los hilos que tienen $M[id] < 0$ (los que tienen sitios de sobra) insertan su id en la lista compartida de Hilos Pesados denotada por HT , (línea 4). Esta inserción se hace a través de exclusión mutua entre los hilos (líneas 2-6). Después todos los hilos se sincronizan mediante un mecanismo de barrera (línea 7). Esta barrera hace que los hilos con $M[id] > 0$ (con sitios faltantes) esperen a que la lista HT sea inicializada por los demás hilos. En el último paso, los hilos con $M[id] > 0$ trabajan utilizando exclusión mutua (líneas 9-23) para tomar sus sitios faltantes de las listas L_S de los hilos en HT . Cuando un hilo id toma los sitios de otro hilo k en HT (líneas 13 y 17), los nuevos valores $M[id]$ y $M[k]$ se actualizan, como se muestra en las líneas 14, 15, 18 y 19. En caso de que un hilo en HT había dado todos sus sitios sobrantes, éste es removido de la lista HT (línea 20).

Una vez que la distribución de los sitios es completada, cada hilo hace crecer su primer cluster generado hasta rellenar por completo su subred y su lista L_S queda vacía.

Asignación de Enlaces

En la parte del sembrado de clusters de sitios cada hilo, también mantiene su lista L_{SC} del mismo modo que se hace en la versión secuencial. Las listas L_{SC} de cada hilo contienen las posiciones en las cuales se insertan los sitios a lo largo de la red porosa; sin embargo los elementos de cada lista L_{SC} están dispersos en todo el espacio de la red porosa.

Debido a las acciones realizadas en la sección anterior las listas L_{SC} pueden estar en desbalance, por lo que es necesario distribuir los elementos entre los hilos. Para distribuir los elementos de las listas L_{SC} se utiliza el mismo algoritmo de la sección anterior (Algoritmo 5), con dos cambios. El primer cambio sustituye en todo el algoritmo L_S por L_{SC} y el segundo cambia la línea 1 por $M[id] = SNS[id] - Size(L_{SC}[id])$.

Una vez que las listas están balanceadas, cada hilo, toma las posiciones de los sitios una a uno de la lista L_{SC} y conecta los tres correspondientes enlaces al sitio (izquierdo, trasero y superior). Cada conexión entre sitio y enlace debe cumplir con el PC y las RG , siguiendo el mismo procedimiento descrito en la sección 5.2.2. Cada sitio es conectado

Algoritmo 5 Algoritmo de redistribución de los sitios entre los hilos

Require: M : Arreglo compartido
Require: HT : Lista compartida de hilos pesados
Require: id : Identificador del hilo

- 1: $M[id] = SNS[id] - SI[id] - Size(L_S[id])$
- 2: **if** $M[id] < 0$ **then**
- 3: **omp critical**
- 4: $INSERTID(id, HT)$
- 5: **end omp critical**
- 6: **end if**
- 7: **omp barrier**
- 8: **if** $M[id] > 0$ **then**
- 9: **omp critical**
- 10: **while** $M[id] > 0$ **do**
- 11: $k = getFirst(HT)$
- 12: **if** $M[id] < |M[k]|$ **then**
- 13: $MOVENSITES(M[id], L_S[id], L_S[k])$
- 14: $M[k] = M[k] + M[id]$
- 15: $M[id] = 0$
- 16: **else**
- 17: $MOVENSITES(M[k], L_S[id], L_S[k])$
- 18: $M[k] = 0$
- 19: $M[id] = M[id] + M[k]$
- 20: $REMOVEID(k, HT)$
- 21: **end if**
- 22: **end while**
- 23: **end omp critical**
- 24: **end if**

a solo tres enlaces porque, como se define en el Capítulo 1, un sitio tiene tres enlaces propios mientras que los otros tres enlaces son compartidos con los sitios vecinos. Lo anterior se hace para evitar la sincronización entre los hilos que posiblemente se necesitaría al conectar los enlaces compartidos (derecho, frontal e inferior). Al finalizar este paso se obtiene una red porosa con posibles violaciones al *PC* y a las *RG*.

6.3.2. Generación de una red porosa válida

Para que la red creada a partir de los pasos descritos en las secciones anteriores sea válida en términos del *PC* y de las *RG*, es necesario aplicar una serie de pasos de *MC* Paralelos Modificados (*MMC-Paralelos*). *MMC-Paralelos* se basa en los *MCs* Paralelos utilizados en la Sección 6.2.2 con dos modificaciones. La primer modificación es que se utiliza la operación de cambio de origen y la segunda es que a cada hilo se le asignan dos subredes. Ambas modificaciones fueron pensadas para que, al igual que un paso de *MC* normal, cada poro pueda ser intercambiado por cualquier otro de la red porosa. Por cada intercambio de sitios o enlaces durante un *MMCs* se verifica que se cumpla tanto con el *PC* y las *RG*. A continuación se detallan los pasos para la eliminación de las violaciones a las *RG*:

1. La red porosa es dividida en $2N$ subredes como se describe en la Sección 6.1.
2. A cada hilo se le asignan aleatoriamente dos subredes de las generadas en el paso anterior
3. Los intercambios se omiten si los poros involucrados pertenecen a las caras externas de sus subredes
4. Cada hilo genera una lista L_{SE} que contiene las posiciones de los sitios que no cumplen con las *RG* en ambas subredes asignadas
5. Si $Size(L_{SE}) > 0$, se toma el primer elementos de la lista L_{SE} que es la posición de un sitio con violaciones geométricas, en otro caso se toma aleatoriamente la posición de un sitio de cualquiera de las dos subredes asignadas, esta posición se etiqueta como $s1$. Una posición($s2$) de un sitio se toma aleatoriamente de cualquiera de las dos subredes asignadas. Posteriormente, $s1$ y $s2$ se intercambian mutuamente sitios o enlaces (ej. intercambiar el enlace derecho de $s1$ por el enlace frontal de $s2$). El intercambio se mantiene solo si el número de violaciones a las *RG* es menor o igual al número de violaciones antes del intercambio, en otro caso el intercambio es rechazado
6. El paso 5 se repite $4((L_x - 2) \cdot (L_y - 2) \cdot (L_z - 2))^3$ veces por cada hilo, después de esto el origen de la red se desplaza de forma aleatoria para que los hilos puedan trabajar en regiones distintas de la red a las que inicialmente se les asignaron
7. Cada hilo calcula el número total de sitios con violaciones a las *GR* en ambas subredes, de existir , incluyendo las caras externas. Cuando el número total de violaciones es igual a cero el algoritmo termina, en otro caso se repite desde el paso 4

6.3.3. Mejoramiento de la isotropía

Una vez que se obtiene una red porosa válida es decir que la red esta libre de violaciones al *PC* y a las *RG*, sin embargo para mejorar la isotropía de la red es necesario aplicar un numero adicional (*K*) de pasos de *MMC*-Paralelos.

Capítulo 7

Resultados

El algoritmo Solución Paralela Aleatoria(*SPA*) descrito en la Sección 6.2 y el algoritmo Solución Paralela Híbrida(*SPH*) descrito en la Sección 6.3 se implementaron haciendo uso del API de OpenMP version 3, las pruebas de estos se realizan en un equipo de cómputo con las siguientes características: 16GB de memoria RAM y un procesador Intel(R) Core(TM) i7 CPU Extreme Edition con 6 cores(12 con hyperthreading). Los respectivos algoritmos secuenciales Solución Aleatoria (*SA*) descrito en la Sección 5.1 y Solución Híbrida(*SH*) descrito en la Sección 5.2 se prueban en el mismo equipo de cómputo haciendo uso de un único procesador(core). Parte de los resultados mostrados a continuación fueron publicados en el artículo "Construction of Porous Networks subjected to Geometric Restrictions by using OpenMP" [16]. Con las soluciones *SA*, *SH*, *SPA* y *SPH* se pueden construir redes porosas que cumplan tanto *PC* y las *RG*; sin embargo el ordenamiento y búsqueda de sitios y enlaces se transforma un mayor número de cálculos para las versiones *SH* y *SPH*.

Los resultados se dividen en cuatro secciones: en la primera sección mostramos los tiempos de ejecución de las versiones secuenciales y paralelas variando el parámetro del traslape (Ω), en la segunda sección mostramos el comportamiento de las versiones paralelas al incrementar el número de hilos utilizados, en la tercera sección mostramos los tiempos de ejecución para la construcción de redes de distintos tamaños, por último en la cuarta sección mostramos de manera gráfica un conjunto de redes porosas antes y después de eliminar las violaciones a las *RG*.

Variación en el traslape

En esta sección mostramos los tiempos de ejecución para la construcción de redes porosas que cumplen con el *PC* y las *RG*, variando el parámetro del traslape (Ω) para las soluciones *SA*, *SH*, *SPA* y *SPH*.

En las Figura 7.1 y 7.2, se muestran los distintos tiempos de construcción para una red porosa de tamaño $L = 100$ utilizando y variando el traslape (Ω), en ambas figuras

podemos observar que conforme el traslape aumenta el tiempo de ejecución también aumenta esto se debe a que tanto en las versiones secuenciales y paralelas el encontrar enlaces que cumplan con las *RG* para un sitio se vuelve más complicado, lo que se traduce en un notable incremento en el número de cálculos.

Comparando las versiones *SA* y *SPA* se puede ver en la Figura 7.1 que *SPA* comienza a tener un mejor rendimiento en términos de tiempo cuando el traslape es cercano o mayor a 0,1 esto se debe a que el número de violaciones a las *RG* aumenta y por consiguiente el número de pasos de Monte Carlo para eliminarlas incrementa y es donde al incrementar el número de hilos comienza a mejorar el rendimiento.

Comparando las versiones *SH* y *SPH* se puede ver en la Figura 7.2 que *SPH* siempre es mejor en términos de tiempo independientemente de valor del traslape esto se debe al procedimiento de sembrado de sitios y asignación de enlaces, lo que permite una mejor distribución del trabajo entre los hilos.

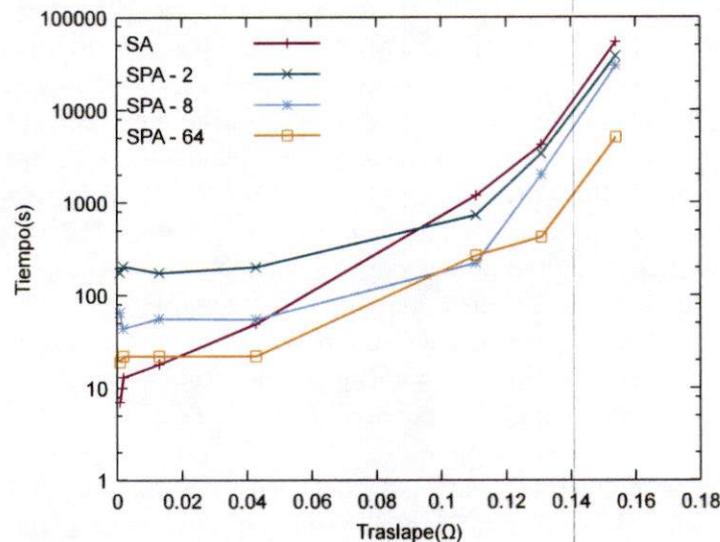


Figura 7.1: Tiempos de ejecución de *SA* y *SPA* utilizando 2, 8 y 64 hilos bajo diferentes valores de Ω (escala logarítmica)

Variación en el número de hilos

En esta sección mostramos el comportamiento de los dos algoritmos paralelos propuestos *SPA* y *SPH*, variando el número número de hilos utilizados para la construcción de una red porosa válida.

En la Figura 7.3, se muestran cuatro casos de comparación entre las soluciones *SPA* y *SPH* en términos del tiempo de construcción de una red porosa de tamaño $L = 100$ con distintos traslapes y variando el número de hilos utilizados para la construcción

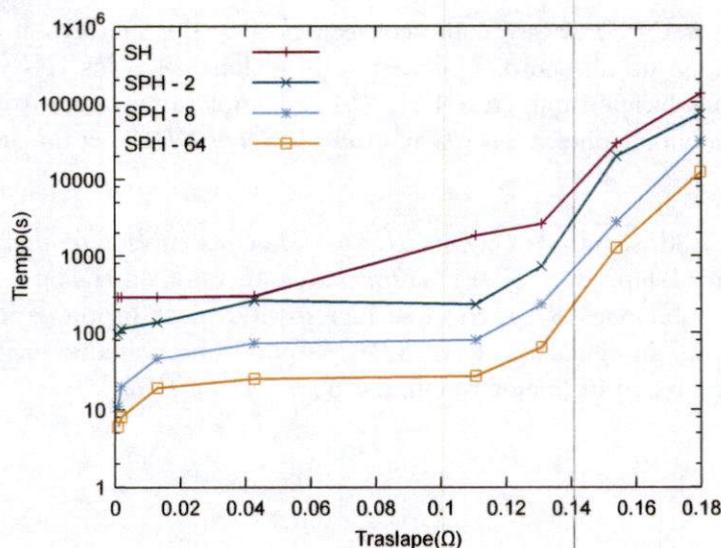


Figura 7.2: Tiempos de ejecución de *SH* y *SPH* utilizando 2, 8 y 64 hilos bajo diferentes valores de Ω (escala logarítmica)

de la red. En cada caso se añade como referencia dos líneas rectas que representan los tiempos de construcción de la red porosa con el traslape especificado para las soluciones *SA* y *SH*.

En la Figura 7.3a, se muestra el primer caso para construcción de una red de tamaño $L = 100$ con un traslape de $\Omega = 0,001908$, la solución *SPH* muestra en general un mejor rendimiento que *SPA* sin embargo muestra un bajo rendimiento respecto a *SA* esto se debe a que al utilizar un traslape tan pequeño el número de violaciones a las *RG* es bajo por lo cual la solución *SPH* consume la mayor parte de su tiempo en la inicialización de la red particularmente en proceso de ordenamiento de sitios y enlaces mientras que *SA* trabaja directamente en la eliminación de las violaciones, lo mismo ocurre con *SH*.

En la Figura 7.3b, se muestra el segundo caso para construcción de una red de tamaño $L = 100$ con un traslape de $\Omega = 0,042809$, podemos ver que *SPA* y *SPH* muestran un compartimento muy similar en el cual a partir del uso de más de 8 hilos mejoran su rendimiento respecto a *SA*, al aumentar el traslape también aumenta el número de violaciones a las *RG* lo que se transforma en mayor trabajo para cada hilo. Para *SPA* en el caso anterior y en el actual (hasta 8 hilos) la mayor parte de tiempo se consumía en la distribución y redistribución de los datos es por eso que se muestra un rendimiento menor que *SA*. Para *SPH* por las mismas causas tiene que un compartimento similar al del caso sin embargo en este caso se obtuvo un rendimiento mejor que el de *SA* utilizando 8 hilos a diferencia de los 32 necesarios en el caso anterior.

En la Figura 7.3c, se muestra el tercer caso para construcción de una red de tamaño $L = 100$ con un traslape de $\Omega = 0,153895$, la solución *SPH* muestra un mejor

rendimiento que *SA* y *SPA* esto como consecuencia de dos factores el primero es que el traslape ocasiono un aumento significativo de violaciones a las *RG* y el segundo es el proceso de inicialización que en *SA* y *SPA* es completamente aleatorio lo que hace que el número de violaciones a las *RG* respecto a *SH* y *SPH* sea mucho mayor.

En la Figura 7.3d, se muestra el cuarto caso para construcción de una red de tamaño $L = 100$ con un traslape de $\Omega = 0,179723$, para este valor de traslape los tiempos de ejecución de las soluciones *SA* y *SPA* se incrementaron de forma exponencial por lo cual se interrumpió su ejecución. Para *SPH* se pudo observar que en todos los casos siempre se mantuvo con un mejor rendimiento respecto a *SH*.

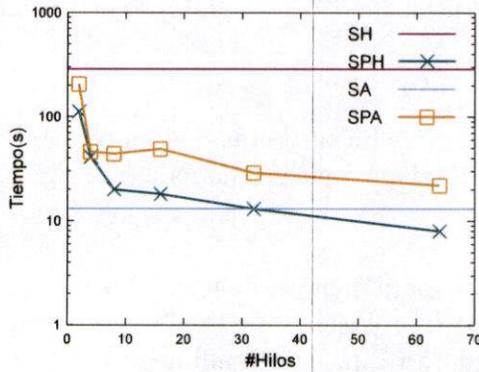
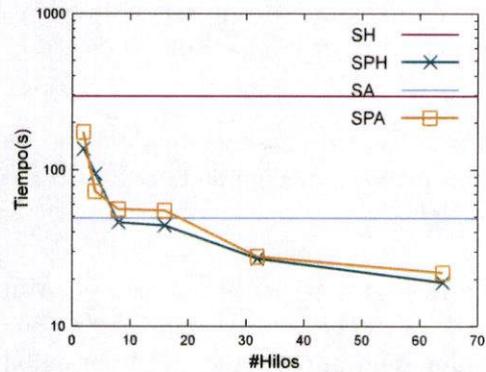
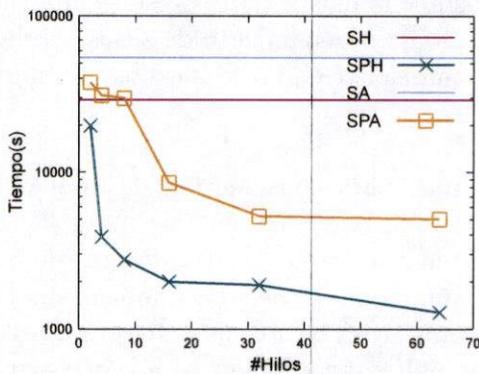
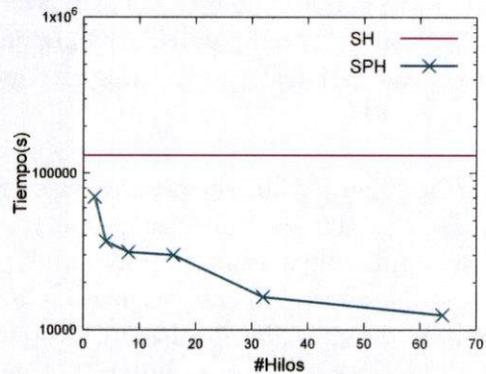
(a) $\Omega = 0,001908$ (b) $\Omega = 0,042809$ (c) $\Omega = 0,153895$ (d) $\Omega = 0,179723$

Figura 7.3: Tiempos de ejecución para *SH*, *SPH*, *SA* y *SPA* para distintos traslapes y variando el número de hilos (escala logarítmica)

Cabe destacar que en todos los casos mostrados en la Figura 7.3, se utilizaron hasta 64 hilos en las pruebas lo cual supera al número cores o hilos de ejecución (hyperthreading) disponibles en el equipo de cómputo donde se realizaron las pruebas, sin embargo esta sobrecarga no afectó de forma negativa el rendimiento de las soluciones *SPA* y *SPH* si no al contrario, esto se debe a dos factores el primero es la naturaleza de los algoritmos utilizados en los cuales al particionar y operar sobre los datos no es necesaria una operación de reducción o unión esto da como resultado que la suma del tiempo de trabajar con conjuntos de datos más pequeños sea menor al tiempo que se necesitaría completar la misma tarea con un conjunto del tamaño de la suma de los conjuntos más pequeños, el segundo factor es la planificación de los hilos por parte del sistema operativo el cual puede manejar un número de hilos superior al número cores o hilos de ejecución (hyperthreading). La mejora en tiempo con sobrecarga se obtuvo de forma constante utilizando hasta 64 hilos, al utilizar un valor más elevado se comenzaron a tener resultados impredecibles.

Variación en el tamaño de la red

En esta sección mostramos el comportamiento en términos de tiempo de ejecución de la solución *SPH* para la construcción de redes porosas válidas de distintos tamaños.

En la Figura 7.4 se muestran los tiempos de ejecución de la *SPH* con 8 hilos para la construcción de redes porosas de distintos tamaños con L desde 100 hasta 350 y variando el traslape Ω . En todos los casos se puede observar que conforme el traslape aumenta el tiempo de ejecución aumenta de forma muy similar para cualquier tamaño.

Analizando estos resultados se pudo ver que el aumento en el tamaño de las redes no afectó de forma negativa al tiempo de ejecución final, el tiempo promedio para la sincronización y balanceo descrito en la sección 5.2 para la *SPH* fue menor a un segundo para redes con L menor o igual a 200 y para redes con L mayor que 200 y menor o igual a 350 fue menor a dos segundos.

Redes porosas

En esta sección mostramos de manera gráfica las redes porosas resultantes antes y después de eliminar las violaciones a las *RG*, dichas redes fueron creadas variando los siguientes parámetros: traslape, número de hilos y el tamaño de la red.

En las Figuras 7.5, 7.6 y 7.7 se muestran algunos ejemplos de redes porosas creadas utilizando *SPH*, en estas imágenes se utilizó el siguiente código de colores: los poros grandes se muestran de color rojo, los poros de tamaño medio de color azul y los poros pequeños en color azul claro(cian). El código de colores nos permite visualizar como están distribuidos los poros a lo largo de la red.

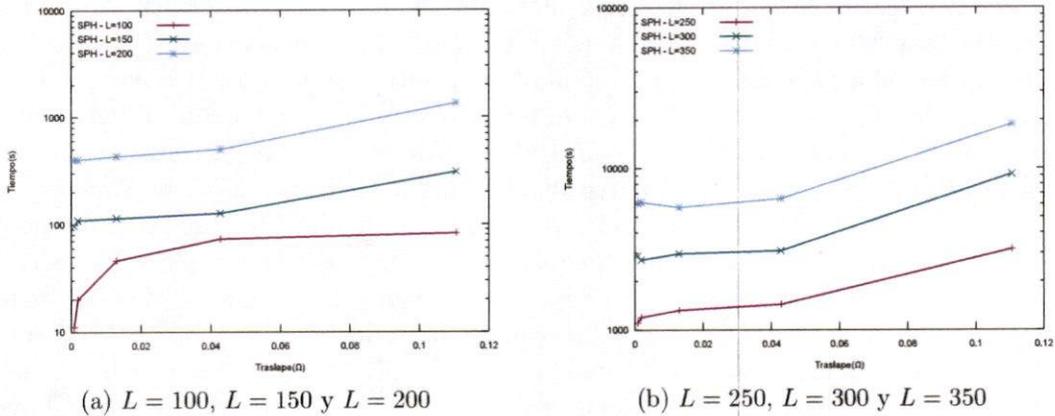


Figura 7.4: Tiempos de ejecución para *SPH* para redes de distinto tamaño y variando el traslape (escala logarítmica)

En la Figura 7.5 se muestran tres ejemplos de redes porosas de tamaño $L = 100$ antes y después de eliminar las violaciones a las *RG* utilizando diferentes traslapes y 8 hilos, se puede observar que en las Figuras 7.5a y 7.5b que una vez que se eliminaron las violaciones a las *RG* en ambos casos se obtuvo una red porosa con una buena isotropía sin embargo en el caso de la Figura 7.5c en donde el traslape fue mayor y se puede apreciar como los poros grandes y medianos tienden a agruparse esto se debe a que a mayor traslape se tiene un número mayor de violaciones a las *RG* y por consiguiente los intercambios de los pasos *MMC-Paralelos* descritos en la Sección 6.3.2 sean más complicados.

En la Figura 7.6 se muestran las redes porosas generadas con tamaño $L = 100$ y con un traslape $\Omega = 0,110658$ antes y después de eliminar las violaciones a las *RG* utilizando 2, 8 y 64 hilos respectivamente. Se puede apreciar que conforme se utiliza un mayor número de hilos la isotropía de las redes mejora esto se debe al cambio de origen en las etapas de sembrado de sitios y en los pasos *MMC-Paralelos* ambos descritos en las secciones 6.3.1 y 6.3.2 respectivamente. A partir de lo anterior podemos ver que la solución *SPH* además de mejorar el tiempo de ejecución se mejora la isotropía de las redes creadas conforme se utiliza un número mayor de hilos.

Por ultimo en la Figura 7.7 se muestran las redes generadas antes y después de eliminar las violaciones a las *RG* con un tamaño L de 150, 250 y 350 respectivamente, en todos los casos se utilizó un el traslape $\Omega = 0,110658$ y se utilizaron 8 hilos para la creación.

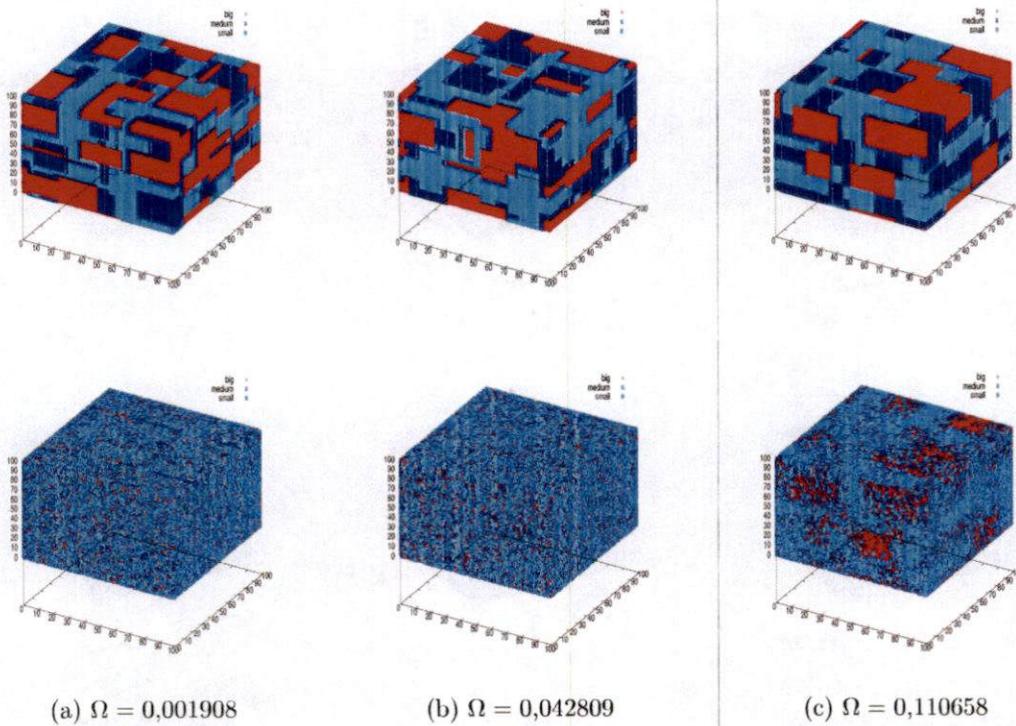


Figura 7.5: Redes porosas antes y después de eliminar las violaciones a las *RG* con diferentes valores de Ω

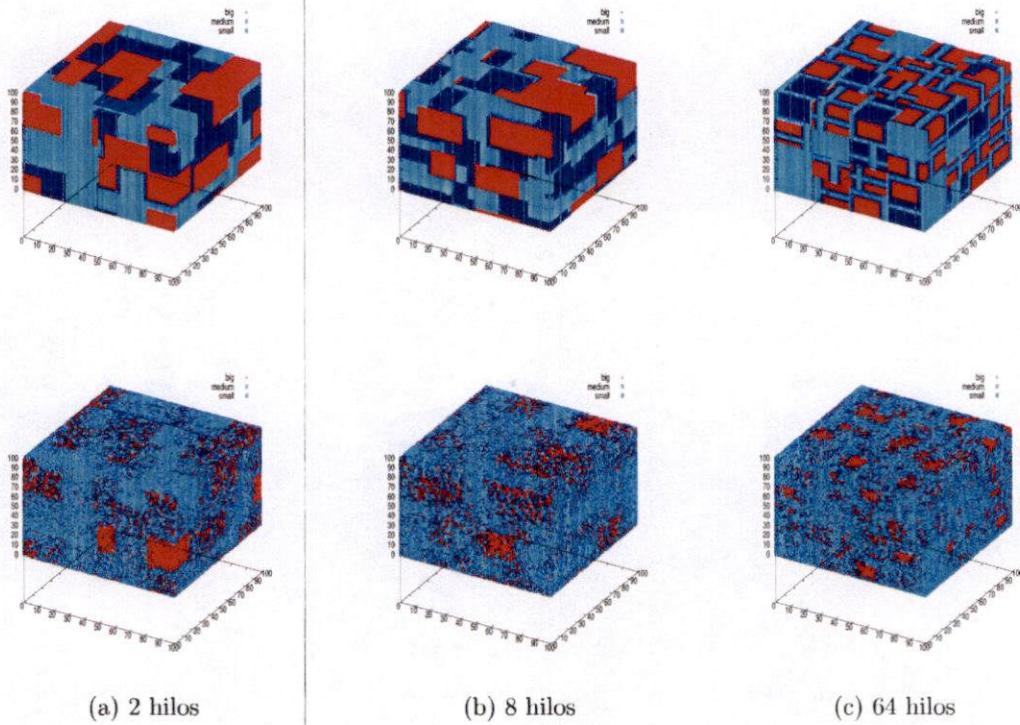


Figura 7.6: Redes porosas antes y después de eliminar las violaciones a las *RG* utilizando un número diferente de hilos

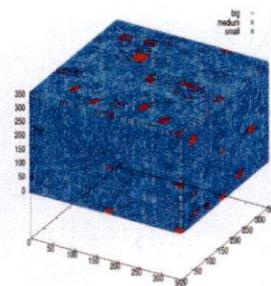
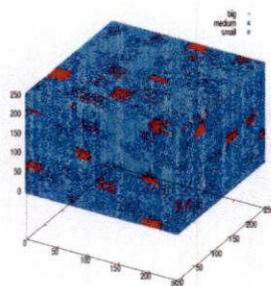
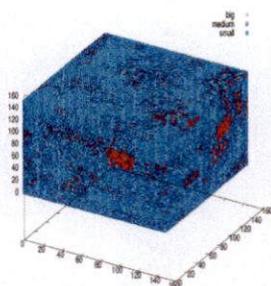
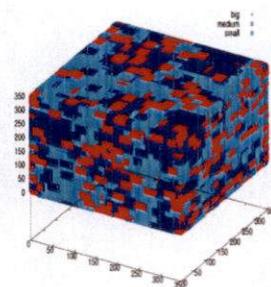
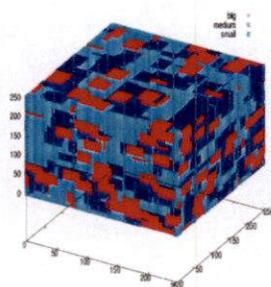
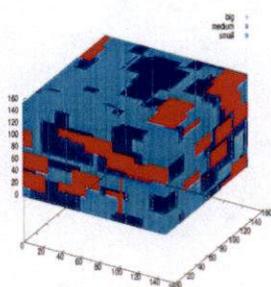
(a) $L = 150$ (b) $L = 250$ (c) $L = 350$

Figura 7.7: Redes porosas antes y después de eliminar las violaciones a las RG de diferentes tamaños

Capítulo 8

Conclusiones y Trabajo Futuro

En este trabajo se presentaron dos soluciones secuenciales y paralelas para la construcción de redes porosas sujetas al cumplimiento de restricciones geométricas. Consideramos que esto es una aportación para el estudio más realista de materiales porosos a través de la simulación por computadora. En las soluciones *SA* y *SH* se pudo ver que para traslapes pequeños la *SA* tuvo un mejor rendimiento en términos de tiempo como consecuencia del bajo número de violaciones a las *RG*, y a que la *SH* realiza una operación de ordenamiento independientemente del traslape utilizado. Sobre las respectivas soluciones paralelas *SPA* y *SPH* se obtuvo que esta última siempre tiene un mejor rendimiento en términos de tiempo respecto a *SH* y también respecto a *SPA*, independientemente del traslape utilizado en las pruebas. Por lo tanto nuestros resultados nos indican que la solución paralela *SPH* es la que dio mejores resultados; por ejemplo utilizando la solución *SPH* con 64 hilos y con una configuración $L = 100$ y $\Omega = 0,110658$ se obtuvo una aceleración en tiempo de hasta 68,48 veces más rápido que su contraparte secuencial *SH*.

Respecto a las redes porosas que se obtuvieron de la *SPH*, estas son las que dieron mejores resultados no solo por la notable ganancia en tiempo sino también por la distribución de los poros a lo largo de la red, sobre esto último cabe destacar que la isotropía de la red mejoraba conforme se utilizaba un mayor número de hilos esto es una consecuencia del propio diseño del algoritmo de la solución *SPH* en el cual se introdujo una distribución de los datos totalmente dinámica es decir que durante la ejecución los hilos iban cambiando su zona de trabajo y además para los pasos *MMC*-Paralelos cada hilo trabajaba en dos zonas diferentes a la vez.

Acerca del trabajo futuro, continuando el trabajo sobre sistemas de memoria compartida una posible mejora sería utilizar operaciones vectoriales para el cálculo y validación de las *RG* a través de tecnologías incluidas en los procesadores modernos como pueden ser *sse3*, *sse4*, *avx*, *avx2*. Sobre modelos híbridos crear una adaptación de *SPA* y *SPH* para que utilice procesadores multi-núcleo y GPU's, en esta aproximación se deberá proponer una nueva política de distribución de datos debido a las restricciones que tiene en términos de manejo de memoria las GPU's. Sobre sistemas con memoria

distribuida se puede plantear el uso de *SPA* y *SPH* a través de MPI, para evitar que la transferencia de datos y sincronización entre los nodos se convierta en un cuello de botella se debe incluir un proceso(hilo) de actualización es decir mientras cada nodo procesa una parte de su subred otro proceso(hilo) independiente actualiza los datos entre los nodos.

Referencias

- [1] S. Cordero, et al., "Review: Site-Bond Network Modeling of Disordered Porous Media," *Particle & Particle Systems Characterization*. vol. 21, pp. 101–116 2004.
- [2] S. Cordero, F. Rojas, J.L. Riccardo, "Simulation of three-dimensional porous networks," *Colloids and Surfaces A: Physicochemical and Engineering Aspects*, vol. 187–188, pp. 425-438, 2001.
- [3] G. Román, F. Rojas, M. Aguilar, S. Cordero, M.A. Castro, "In-silico simulation of porous media: Conception and development of a greedy algorithm," *Microporous and Mesoporous Materials*, vol. 137, pp. 18-31, 2011.
- [4] J. Matadamas, G. Román-Alonso, F. Rojas, M. Castro, A. Boukerche, M. Aguilar, S. Cordero, "Parallel Simulation of Pore Networks Using Multicore CPUs", *IEEE Transactions on Computers*, 2012.
- [5] "Refinements of the Twofold Description of Porous Media", Mayagoitia V., Rojas F., Kornhauser I., Zgrablich G., Faccio R.J., Gilot B., and Guigliion C., *Langmuir*, 12, 211-216, 1996.
- [6] B. Chapman, G. Jost, R. van der Pas, "Using OpenMP: Portable Shared Memory Parallel Programming," *The MIT Press*, pp. 384, 2007.
- [7] C.H. Moreno, F. Rojas, G. Román, S. Cordero, M.A. Castro, M. Aguilar, "A Parallel Simulator for Mercury (Hg) Porosimetry," M. Ropo et al. (Eds.): *EuroPVM/MPI 2009*. LNCS, Springer, vol. 5759, pp. 294–304, 2009.
- [8] M. Sahimi, "Flow phenomena in rocks: from continuum models to fractals, percolation, cellular automata, and simulated annealing," *Rev. Mod. Phys.* 1993, 65(4), 1393-1534.
- [9] U. Gil-Cruz, M. A. Balderas-Altamirano, S. Cordero-Sánchez, "Textural study of simulated dimorphic porous substrates," *Colloids Surf. A*, 2010, 357(1-3), 8492UTF{2013}90.
- [10] M.D. Kalugin, and A.V. Teplukhin, "Parallel Monte Carlo study on caffeine-DNA interaction in aqueous solution", *IPDPS*, pp.1-8, 2009 *IEEE International Symposium on Parallel & Distributed IP Processing*, 2009.

- [11] V. Mayagoitia, F. Rojas, I. Kornhauser and H. Pérez-Aguilar. "Modeling of Porous Media and Surface Structures: Their True Essence as Networks," *Langmuir*. vol. 13, pp. 1327-1331 1997.
- [12] T. Sterling. "Beowulf Cluster Computing with Linux," MIT Press, Cambridge, 2001.
- [13] David R. Butenhof, "Programming With Posix Threads," Addison-Wesley, 1997.
- [14] H. Chung, C. Chang, H. Hsiao and Y. Chao, "The Load Rebalancing Problem in Distributed File Systems," *Cluster Computing (CLUSTER)*, IEEE International Conference on, pp.117,125, 24-28 Sept. 2012.
- [15] M. Newman and G. Barkema, "Monte Carlo Methods in Statistical Physics," Oxford University Press, Chap. 2, pp. 31-44, 2007.
- [16] A. González-Méndez, G. Román-Alonso, F. Rojas-González, M.A. Castro-García, M. Aguilar-Cornejo, and S. Cordero-Sánchez; "Construction of Porous Networks subjected to Geometric Restrictions by using OpenMP"; IEEE 28th International Parallel & Distributed Processing Symposium Workshops, Phoenix, USA, pp. 1189-1197, 2014.
- [17] Fernando Rojas-González, Graciela Román-Alonso, Salomón Cordero-Sánchez, Miguel Alfonso Castro-García, Manuel Aguilar-Cornejo and Jorge Matadamas Hernández; *Book Comprehensive Guide for Mesoporous Materials*"; Chapter : "On The Conception and Assessment of Mesopore Networks: Development of Computer Algorithms"; Nova Science Publishers, Inc. vol. 3, pp. 343-370, 2015, ISBN: 978-1-63463-318-5, 2015.
- [18] S. Cordero, F. Rojas, and J.L. Riccardo; "Simulation of three-dimensional porous networks *Colloids and Surfaces A: Physicochemical and Engineering Aspects*", 187-188 (2001), pp.425- 438, 2001.