UNIVERSIDAD CATÓLICA
"NUESTRA SEÑORA DE LA ASUNCIÓN"

FACULTAD DE CIENCIAS Y TECNOLOGÍA

CARRERA DE INGENIERÍA INFORMÁTICA

PROYECTO FINAL DE TESIS

PARTICIÓN DE SISTEMAS DE ECUACIONES PARA SU RESOLUCIÓN EN UN AMBIENTE COMPUTACIONAL DISTRIBUIDO

Diana Beatriz Benítez Cáceres

Febrero de 1997

Este trabajo está dedicado a mis padres.

Espero poder retribuirles en alguna medida su esfuerzo, su apoyo, su comprensión y sobre todo, el amor que he recibido de ellos desde siempre.

¡muchas gracias !

Agradecimientos

A mi hermano Víctor Manuel que supo apoyarme en todo momento, inclusive muchas veces sacrificando sus propias prioridades.

Al Profesor Benjamín Barán, excelente profesional y por sobre todo, excelente persona. Sin su constante apoyo y orientación este proyecto no hubiera podido concretarse. Su excepcional capacidad de superación frente a los problemas ha inyectado vida y dinamismo al quehacer diario de las personas que tenemos la gran suerte de pertenecer a su grupo de investigación.

Al Profesor Gerónimo Bellasai, por inculcarnos que las herramientas que hemos adquirido en nuestro paso a través de nuestra querida facultad nos sirvan para forjarnos una vida digna, honesta y plena, al servicio de nuestras familias y nuestro país.

Al Ing. Rodrigo Ramos, gran compañero y mejor amigo, quien supo compartir a lo largo de este tiempo todos los sinsabores y todos los triunfos con la mejor predisposición de ánimo y un gran profesionalismo.

A todos mis profesores, compañeros y amigos de la Facultad de Ingeniería Informática con quienes he compartido estos años de formación.

Finalmente, quiero agradecer a Dios y a la Virgen María todas las bendiciones recibidas a lo largo de mi camino y por permitirme llegar a esta etapa de mi carrera.

TABLA DE CONTENIDO

Lista de Figuras

Obs.: Algunos gráficos no se encuentran en la versión digital.

Lista de Tablas

CAPITULO 1: INTRODUCCION

1.1 Consideraciones iniciales

La amplia gama de problemas de ingeniería existentes en la actualidad requiere a menudo de la resolución de grandes sistemas de ecuaciones. La resolución de los mismos exige cada vez más un enorme esfuerzo computacional, debido a diversas razones, como las grandes dimensiones de los sistemas, la exigencia de modelos matemáticos más detallados y la necesidad de respuestas en tiempo real.

Es reconocido el increíble nivel alcanzado en lo que se refiere a la velocidad de trabajo de los procesadores. Es así que hoy en día se está llegando al límite teórico de velocidad de los procesadores construidos en base a la tecnología de los semiconductores, pudiéndose concluir de esto que el aumento en la velocidad de procesamiento de estos procesadores no podrá continuar sostenidamente en los próximos años, como muestra la figura 1.1, donde se observa el avance histórico de la velocidad de reloj de los microprocesadores fabricados por INTEL [11].

Por otro lado, la velocidad de comunicación con que operan las modernas redes de computadoras está aumentando a un ritmo sostenido, como puede apreciarse en la figura 1.2, donde se muestra la evolución de la misma a través de los últimos 18 años [8]. De acuerdo a las tendencias apreciadas en estas 2 figuras y para un mismo problema, la relación tiempo de procesamiento/tiempo de comunicación será cada vez menor, disminuyendo el costo computacional del envío de datos a través de una red de comunicaciones, surgiendo así la computación paralela/distribuida como la alternativa más efectiva para la resolución de problemas de dimensiones y complejidad creciente.

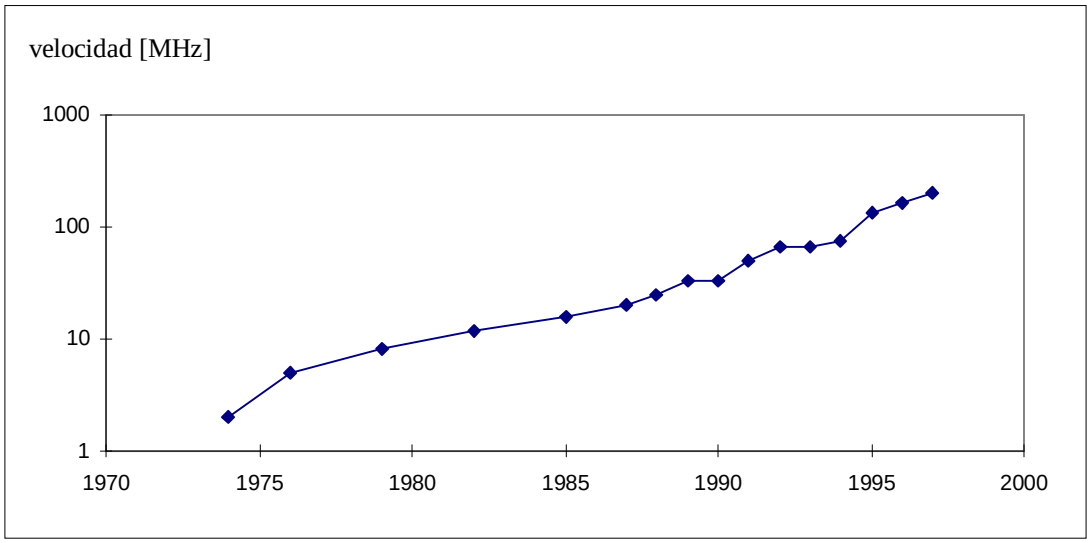


Figura 1.1 : Evolución de la velocidad de procesamiento.

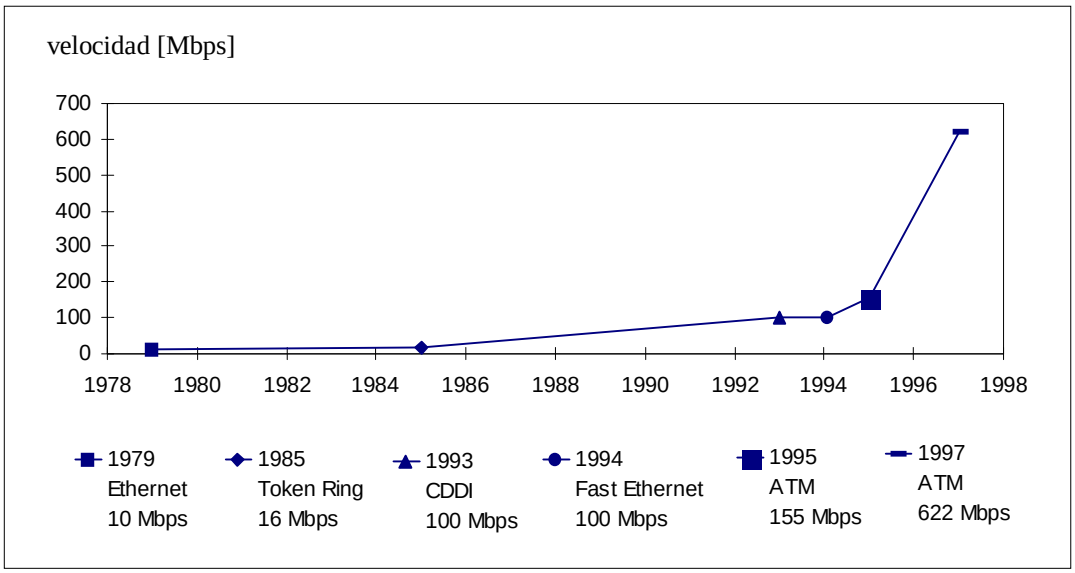


Figura 1.2 : Evolución de la velocidad de comunicación.

1.2 Procesamiento paralelo

El procesamiento paralelo consiste en ejecutar *simultáneamente* varias tareas de un problema de gran dimensión o complejidad. Esta tecnología ha emergido como una importante opción en la computación moderna de alta *performance*, siendo los últimos

años testigos de una aceptación creciente del procesamiento paralelo, tanto para computación científica de alto desempeño como para propósitos más generales como la administración y consulta de grandes bases de datos. En consecuencia a esta tendencia, estas plataformas de trabajo están captando cada vez más el interés del mercado y la atención de los investigadores, como una interesante opción que permite el procesamiento de grandes volúmenes de información y la resolución de problemas computacionales de gran complejidad, que de ser resueltos en un solo procesador exigirían mayor tiempo y posiblemente requerirían de mayores inversiones.

La plataforma computacional necesaria para la aplicación del procesamiento paralelo consiste en un sistema computacional distribuido/paralelo, es decir, un conjunto de procesadores interconectados entre sí por una red de comunicación. Pudiendo utilizarse para este fin inclusive microprocesadores de muy bajo costo, es evidente las ventajas económicas que acarrea esta implementación, teniendo en cuenta el gran parque de computadoras personales (PC) disponibles en la actualidad.

Las técnicas de procesamiento paralelo han sido aplicadas con éxito en diversas áreas de la ciencia y la ingeniería, como [19] hidrodinámica computacional, programas de sismografía, óptica física, ingeniería genética, medicina (tomografía cerebral computada), diseño de motores, mecánica cuántica, etc.

La mayoría de estas aplicaciones utilizan modelos matemáticos que implican el procesamiento de gran cantidad de datos. En este contexto, la aplicación del procesamiento paralelo a la resolución de problemas presenta excelentes perspectivas, como puede apreciarse en recientes trabajos de investigación realizados sobre el tema [2, 4, 5].

1.3 Descomposición de problemas

Para que un problema particular sea resuelto utilizando procesamiento paralelo, es necesario que dicho problema sea “paralelizable”, es decir, que las características de los métodos implementados en la resolución del mismo permitan la distribución de diversas tareas a los procesadores componentes del sistema distribuido, de manera tal a resolver el problema en su conjunto.

De los numerosos problemas paralelizables que se conocen, nos ocuparemos preferentemente de los problemas de ingeniería que se plantean como un sistema (no necesariamente lineal) de ecuaciones, que por lo general son resueltos utilizando métodos iterativos. Para paralelizar estos métodos, nacen los métodos bloque-iterativos, que consisten en asignar a los procesadores del sistema distribuido distintos grupos de ecuaciones a ser resueltos localmente. Los resultados obtenidos localmente por cada uno de ellos son transmitidos a los demás procesadores del sistema a través de la red de comunicación, avanzando el conjunto hacia la solución global del sistema de ecuaciones de una forma típicamente iterativa.

Para que la implementación de estos métodos sea computacionalmente eficiente, es importante descomponer el sistema de ecuaciones de forma a promover el adecuado “mapeamiento” del problema para la arquitectura paralela; esto es, debe descomponerse el sistema de ecuaciones en subsistemas tales que la implementación del algoritmo de resolución sea computacionalmente eficiente.

La literatura presenta diversos trabajos sobre métodos de partición que descomponen un determinado sistema de ecuaciones utilizando diversos criterios. Desafortunadamente, la mayoría de estos métodos no poseen la eficiencia requerida con relación al desempeño de los métodos de solución que resuelven el sistema descompuesto. Surge así la motivación de este trabajo, que consiste en desarrollar un método de

descomposición de sistemas de ecuaciones que permita su resolución eficiente utilizando procesamiento paralelo en un sistema distribuido heterogéneo. Una resolución eficiente es aquella que utiliza un menor tiempo computacional en llegar a la solución, buscando balancear de manera adecuada la carga computacional a ser repartida a los diversos procesadores del sistema computacional.

De esto último se puede concluir que, considerando la posibilidad de operar en un ambiente computacional compuesto por procesadores de diversas *performances*, interconectados por un sistema de comunicación, la descomposición de la red estará condicionada por dos factores:

a) El *balanceamiento de carga computacional*. Este factor determina las dimensiones de los subsistemas asignados a cada procesador, que deberán estar en proporción directa con la capacidad relativa de procesamiento de cada uno de los procesadores utilizados en la resolución de un problema;

b) El *grado de acoplamiento* que puedan tener los subsistemas entre sí, lo que determina a su vez la dependencia entre las incógnitas del sistema de ecuaciones y por consiguiente, influye en la convergencia del algoritmo [30].

En conclusión, el problema que se plantea, dado un sistema distribuido con cierto número de procesadores, consiste en descomponer un sistema de ecuaciones en varios subsistemas menores, de forma tal que a cada procesador se le asigne un subsistema de dimensión proporcional a su performance, y que la dependencia entre las variables actualizadas por cada uno de los procesadores facilite la convergencia del algoritmo a ser utilizado.

1.4 Revisión bibliográfica

La partición de sistemas de ecuaciones no es un tema de introducción reciente, encontrándose los primeros trabajos sobre el tema a finales de la década del 60, cuando Carré [9] ya afirmó que el acoplamiento entre los distintos subsistemas influye en la convergencia de los algoritmos iterativos utilizados en la resolución de sistemas eléctricos. Desde entonces, se sucedieron diversas publicaciones, las cuales presentan características variadas de acuerdo a las diferentes aplicaciones a las que fueron destinadas. De estos trabajos, varios buscan la eficiencia de técnicas de resolución bien específicas como, por ejemplo, la técnica diacóptica [13, 28], eliminación de Gauss [17] o la programación no lineal [22]. Otros trabajos buscan la identificación de "clusters" [21] o la descomposición de sistemas de ecuaciones para posibilitar la solución computacional de redes eléctricas de gran porte [1], sin tener en cuenta la existencia de una red heterogénea. Puede verse así que los métodos precedentes no están directamente relacionados con la aplicación del procesamiento paralelo, aún cuando utilicen sistemas de ecuaciones descompuestos.

Aun así, se dispone en la actualidad de algunos métodos volcados a la aplicación de la computación paralela, presentando algunos de ellos características muy interesantes [12, 15, 20, 32]. Pero, a pesar de todo, no se encuentra en la mayoría de ellos un análisis de los aspectos básicos que favorecen el desempeño de los métodos de solución que irán a utilizar el sistema de ecuaciones descompuesto. Entre los trabajos que incluyen dicho enfoque resaltan de manera especial dos: la *Descomposición* ε [24 - 26] y el *Método de la Semilla* [29, 30]. El primero de ellos, aún cuando es el más implementado en la actualidad, posee la limitación de no poder ejercer un control efectivo sobre el número de subsistemas en el cual el sistema de ecuaciones es descompuesto ni sobre las dimensiones de los mismos. El segundo método, el *Método de la Semilla*, fue introducido en 1992 por Vale et al. [29]. Una vez asociado el sistema de ecuaciones a un grafo donde los nodos son las incógnitas y las ramas o lados son los coeficientes, este método se basa en la

identificación de los centros de aglutinamiento de incógnitas, buscando separar el sistema de ecuaciones en los lugares donde el acoplamiento entre las mismas es menor. Pero nuevamente se presenta la imposibilidad de ejercer dominio sobre las dimensiones de los subsistemas, al suponerse en dicho trabajo que el sistema sería resuelto utilizando un sistema distribuido homogéneo, es decir, los subsistemas resultan siempre de igual dimensión. Inspirado en este último trabajo, surge la presente propuesta que propone una nueva metodología de partición que permite generar subsistemas con las dimensiones requeridas por el usuario.

1.5 Objetivos, metodología y organización del presente trabajo

Teniendo en cuenta la finalidad principal del presente trabajo, los objetivos generales del mismo se establecieron como sigue:

- Desarrollar un método computacional automático, con posibilidad de interacción con el usuario, para descomponer un sistema de ecuaciones de forma tal que su resolución paralela en un sistema distribuido heterogéneo sea eficiente.
- Demostrar experimentalmente las ventajas del método propuesto con respecto a los métodos de partición disponibles en la actualidad.

En la prosecución de los objetivos citados, la metodología de trabajo se basó en los siguientes puntos:

- Análisis de los métodos de partición disponibles.
- Evaluación experimental de dichos métodos.
- Proposición de un método alternativo.
- Evaluación experimental del método propuesto, utilizando sistemas de ecuaciones correspondientes a sistemas eléctricos paradigmas de la IEEE.

El trabajo fue dividido en 8 capítulos. La formulación matemática del problema de la partición de sistemas de ecuaciones se encuentra en el Capítulo 2, con un apartado dedicado al repaso de las características de los métodos bloque-iterativos.

En el Capítulo 3 se exponen los principales métodos de partición de sistemas de ecuaciones disponibles en la actualidad: la *Descomposición* ε y el *Método de la Semilla*.

La exposición pormenorizada del método de partición propuesto se encuentra en el Capítulo 4. Se detallan cada una de las etapas de que consta el mismo y los parámetros que influyen en cada una de estas etapas.

A manera de ejemplo ilustrativo, en el Capítulo 5 se procede a la aplicación del método propuesto a un problema ejemplo diseñado para el efecto. Las distintas etapas del método, así como las distintas peculiaridades que pueden presentarse en su aplicación, son analizadas en forma detallada.

En el Capítulo 6 se presenta el diseño del programa generador de descomposiciones implementado, incluyendo diagramas de flujo, diseño de pantallas y pseudocódigo.

Los estudios experimentales realizados con el fin de verificar la validez del método propuesto se presentan en el Capítulo 7, donde se resolvieron sistemas eléctricos paradigmas proveídos por le IEEE. Se describe además en forma detallada la plataforma computacional distribuida utilizada para realizar los estudios.

En el Capítulo 8 se presentan las conclusiones que han podido ser obtenidas del presente trabajo.

En forma anexa, el Apéndice A presenta el código en lenguaje C++ de las diversas etapas del método. El Apéndice B contiene una breve explicación del problema de Flujo de Potencia Eléctrico y de su resolución distribuida, mientras el Apéndice C presenta el

artículo publicado en el marco de la XXII Conferencia Latinoamericana de Informática. En dicho artículo se demuestra la eficacia del método aplicado a la partición de un sistema de ecuaciones para su resolución en un ambiente distribuido.

CAPITULO 2: FORMULACION MATEMATICA DEL PROBLEMA

En este capítulo se formaliza matemáticamente el problema de la partición de un sistema de ecuaciones para su resolución en un sistema distribuido. Inicialmente se presenta una breve recapitulación sobre los métodos de resolución bloque-iterativos y luego se explica brevemente ciertos conceptos que serán utilizados a lo largo del trabajo.

2.1 Métodos bloque iterativos para sistemas lineales

La presente sección describe cómo un sistema lineal de n ecuaciones con n incógnitas se resuelve utilizando un sistema distribuido compuesto por $p \leq n$ procesadores.

Sea un sistema lineal y no singular de ecuaciones:

$$\mathbf{Ax} = \mathbf{b}, \qquad \mathbf{A} \in \Re^{n \times n}, \qquad \mathbf{x},\ \mathbf{b} \in \Re^{n} \tag{2.1}$$

El sistema puede ser descompuesto como :

$$\begin{bmatrix} \mathbf{A}_{11} \cdots \mathbf{A}_{1p} \\ \vdots \quad \ddots \quad \vdots \\ \mathbf{A}_{p1} \cdots \mathbf{A}_{pp} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_p \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_p \end{bmatrix}, \mathbf{A}_{ij} \in \Re^{n_i \times n_j} \ , \qquad i, j \in \{1, \ldots, p\} \tag{2.2}$$

donde:

$$\mathbf{x}_i, \mathbf{b}_i \in \Re^{n_i} \qquad ; \qquad n = \sum_{i=1}^{p} n_i$$

Desdoblando la matriz $\mathbf{A}$ según $\mathbf{A} = \mathbf{S} - \mathbf{T}$, de forma tal que $\mathbf{S}$ sea bloque diagonal e invertible por bloques, se obtiene

$$\mathbf{Sx} - \mathbf{Tx} = \mathbf{b} \tag{2.3}$$

$$\mathbf{x} = \mathbf{S}^{-1}(\mathbf{b} + \mathbf{T}\mathbf{x}) \tag{2.4}$$

de donde se deriva el método bloque-iterativo de Jacobi:

$$\mathbf{x}(k+1) = \mathbf{S}^{-1}(\mathbf{b} + \mathbf{T}\mathbf{x}(k)) = \begin{bmatrix} \mathbf{S}_{11} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{S}_{pp} \end{bmatrix}^{-1} \left\{ \begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_p \end{bmatrix} + \begin{bmatrix} \mathbf{T}_{11} & \cdots & \mathbf{T}_{1p} \\ \vdots & \ddots & \vdots \\ \mathbf{T}_{p1} & \cdots & \mathbf{T}_{pp} \end{bmatrix} \mathbf{x}(k) \right\} \tag{2.5}$$

donde k es el número de iteración.

(2.5) puede implementarse haciendo que cada procesador i de un sistema distribuido de p procesadores actualice $\mathbf{x}_i$ conforme a:

$$\mathbf{x}_i(k+1) = \mathbf{S}_{ii}^{-1} \left(\sum_{j=1}^{p} \mathbf{T}_{ij}\, \mathbf{x}_j(k) + \mathbf{b}_i \right), \quad \forall i \in \{1, \ldots, p\} \tag{2.6.a}$$

De esta forma, el problema (2.1) puede ser resuelto con p procesadores de forma tal que en una iteración k, cada procesador i solo actualiza su incógnita local x_i conforme a (2.6.a), comunicando el valor actualizado a los demás procesadores del sistema distribuido.

El objetivo del presente trabajo es encontrar la partición de la matriz **A** en las correspondientes submatrices $\mathbf{A}_{ij}$ de forma a asignar a cada procesador un subproblema proporcional a su capacidad de procesamiento (performance relativa), además de buscar obtener la mejor convergencia posible, es decir, que el método de resolución a ser utilizado emplee un número reducido de iteraciones hasta llegar a la solución global del problema.

2.2.- Sincronismo y asincronismo

En una resolución bloque-iterativa, cada procesador actualiza su incógnita local $\mathbf{x}_i$ utilizando los resultados obtenidos en la iteración anterior por los diversos procesadores del sistema, que les fueron comunicados a través del mecanismo de comunicación (por ejemplo la red). Este proceso continúa hasta que todo el conjunto llegue a la solución global del problema.

La comunicación de los resultados parciales puede ser realizada de forma síncrona o asíncrona [6]:

Comunicación síncrona: ocurre cuando cada procesador espera hasta disponer de todos los resultados parciales de la iteración anterior, calculados por los demás procesadores del sistema, antes de calcular su siguiente iteración. Al tiempo transcurrido entre la terminación de su iteración y el inicio de la siguiente iteración se denomina "tiempo de sincronización", y es utilizado para el envío de resultados parciales entre procesadores.

En consecuencia, la implementación síncrona de (2.6.a) puede ser representada para un ejemplo de dos procesadores conforme a la figura 2.1.a, donde cada procesador debe esperar los datos de la iteración anterior antes de iniciar la siguiente iteración. La figura 2.1.a. ilustra los tiempos muertos o de "sincronización".

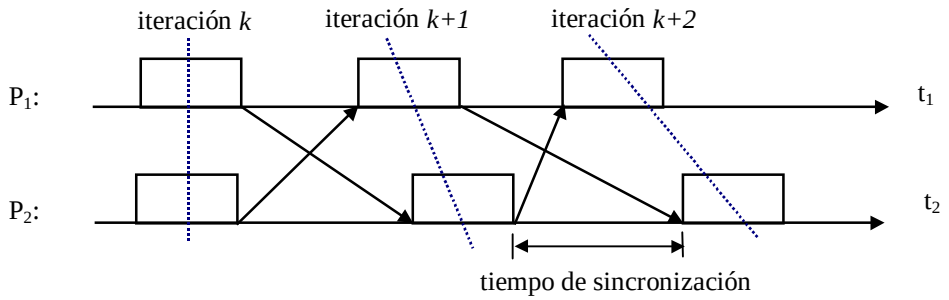


Figura 2.1.a: Resolución síncrona

Comunicación as rocesador utiliza los resultados parciales más recientes que ha recibido de los demás procesadores al momento de iniciar

una iteración. La comunicación asíncrona contempla la posibilidad de que un procesador más rápido trabaje a su propia velocidad, sin depender de las velocidades relativas de los demás procesadores. Esto conlleva diferentes números de iteraciones por procesador, según sea la capacidad de procesamiento del mismo y la dimensión del problema local a él asociado.

Si bien la comunicación asíncrona “ahorra” tiempo de sincronización, se resalta la dificultad de lograr un control estricto de las actividades realizadas por cada uno de los procesadores en dicho contexto, debido a la complejidad del proceso de comunicación implicado. Por esto, es más difícil obtener la convergencia de los métodos de resolución bloque-iterativos en la resolución asíncrona que en la correspondiente resolución síncrona, aunque una vez obtenida, el asincronismo generalmente alcanza la solución en menores tiempos de procesamiento [3].

Esta característica de las implementaciones asíncronas trae consigo la posibilidad de utilizar valores no totalmente actualizados de algunas variables calculadas en otros procesadores. Así, en un contexto asíncrono la ecuación (2.6.a) debería re-escribirse como:

$$\mathbf{x}_i(k+1) = \mathbf{S}_{ii}^{-1} \left(\sum_{j=1}^{p} \mathbf{T}_{ij}\, \mathbf{x}_j^i(k) + \mathbf{b}_i \right), \quad \forall i \in \{1, \ldots, p\} \qquad (2.6.b)$$

donde $\mathbf{x}_j^i(k)$ representa el valor más actualizado de la variable $\mathbf{x}_j$, disponible en el procesador *i* al momento de iniciar su iteración *k*.

Entonces, una implementación asíncrona de (2.6.b) para dos procesadores puede ser representada por la figura 2.1.b, donde ambos procesadores no necesitan esperar recibir los datos de la iteración anterior para iniciar la siguiente iteración.

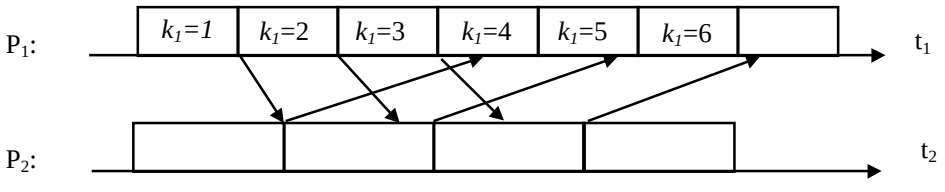


Figura 2.1.b: Resolución asíncrona

2.3.- Generalización del método iterativo

Se considera a continuación un sistema de *n* ecuaciones algebraicas con *n* incógnitas posiblemente no lineal, dado por:

$$\Phi(\mathbf{x}) = \mathbf{0}, \qquad \mathbf{x} \in \Re^n, \qquad \Phi: D \to \Re^n \tag{2.7}$$

definido en un dominio $D \subset \Re^n$; se desea resolver (2.7) utilizando un sistema distribuido con *p* procesadores, donde $n \geq p$. Para esto, se debe partir el problema de manera tal que cada procesador "resuelva" solamente una parte (o subproblema) del sistema completo, comunicando los resultados parciales a los otros procesadores, para finalmente resolver en conjunto el problema global. Se introducen a continuación la notación y algunas definiciones a ser utilizadas.

Sea la Descomposición Cartesiana de $\Re^n$ dada por:

$$\Re^n = \Re^{n_1} \times \cdots \times \Re^{n_p}, \qquad n_1 + \cdots + n_p = n \tag{2.8}$$

con $D \subset \Re^n$ un dominio tal que

$$D = D_1 \times \cdots \times D_p, \qquad D_i \subset \Re^{n_i}, \qquad \forall i \in \{1,\ldots,p\} \tag{2.9}$$

El vector incógnita **x** puede ser descompuesto conforme a:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_p \end{bmatrix}, \qquad \mathbf{x}_i \in D_i, \qquad \forall\ i \in \{1,\ldots,p\} \tag{2.10}$$

Análogamente, $\Phi(\mathbf{x})$ puede descomponerse como:

$$\underset{\sim}{\Phi}(\mathbf{x}) = \begin{bmatrix} \underset{\sim}{\Phi}_1(\mathbf{x}) \\ \vdots \\ \underset{\sim}{\Phi}_p(\mathbf{x}) \end{bmatrix}, \qquad \underset{\sim}{\Phi}_i : D \to \Re^{n_i} \tag{2.11}$$

Consecuentemente, la ecuación a ser resuelta en cada procesador *i* (problema local), estará dada por:

$$\underset{\sim}{\Phi}_i(\mathbf{x}) = \mathbf{0}, \qquad \forall i \in \{1,\ldots,p\} \tag{2.12}$$

NOTA 1: *La partición adecuada de la función* $\underset{\sim}{\Phi}(\mathbf{x})$ *en p funciones* $\underset{\sim}{\Phi}_i(\mathbf{x})$ *es el objetivo central del presente trabajo.*

Para resolver el problema será utilizado un algoritmo iterativo de la forma

$$\mathbf{x}(k+1) = \mathbf{G}(\mathbf{x}(k)) \tag{2.13}$$

donde *k* representa la iteración. Por lo tanto, en forma análoga a (2.10) y (2.11):

$$\mathbf{G}(\mathbf{x}) = \begin{bmatrix} \mathbf{G}_1(\mathbf{x}) \\ \vdots \\ \mathbf{G}_p(\mathbf{x}) \end{bmatrix}, \qquad \mathbf{G}_i(\mathbf{x}) : D \to D_i, \qquad \forall i \in \{1,\ldots,p\} \tag{2.14}$$

La implementación síncrona del algoritmo (2.13) para el procesador *i* será entonces:

$$\mathbf{x}_i(k+1) = \mathbf{G}_i(\mathbf{x}(k)) \tag{2.15}$$

Utilizando la notación dada en [14] podemos escribir:

$$\mathbf{x}^{k+1} = \mathbf{H}(\mathbf{x}^k, k)\,\mathbf{x}^k \tag{2.16}$$

donde la *función iteración* $\mathbf{H}(\mathbf{x}^k, k)$ puede ser no lineal y variante en el tiempo.

A continuación, se harán las siguientes hipótesis:

Hipótesis 1 (Unicidad de la solución)

Dado el sistema de ecuaciones (2.7) que puede ser reescrito como (2.12) , definido en un dominio cerrado $D \subset \mathbb{R}^n$ *que satisface (2.9), existe un operador* **G** *de la forma (2.14), tal que* $\mathbf{G}(D) \subset D$ *y adicionalmente D contiene un y solamente un punto fijo*

$$\mathbf{x}^* = \begin{bmatrix} \mathbf{x}_1^* \\ \vdots \\ \mathbf{x}_p^* \end{bmatrix}, \qquad \mathbf{x}_i^* \in D_i , \qquad \forall\, i \in \{1,\ldots,p\} \tag{2.17}$$

del operador **G**. *El punto fijo* $\mathbf{x}^*$ *es a su vez solución de (2.7).*

Consecuentemente podemos escribir que:

$$\mathbf{x}_i^* = \mathbf{G}_i(\mathbf{x}^*) , \qquad \forall\, i \in \{1,\ldots,p\} \tag{2.18}$$

y

$$\mathbf{F}_i(\mathbf{x}^*) = \mathbf{0} , \qquad \forall\, i \in \{1,\ldots,p\} \tag{2.19}$$

Considerando un contexto asíncrono en el cual los procesadores trabajan sin barreras de sincronización, transmitiendo los valores calculados y recibiendo los valores que le van llegando, se denota como $\mathbf{x}_j^i(k)$ al valor de $\mathbf{x}_j$ enviado desde el procesador *j*, y disponible en el procesador *i* al iniciar su iteración local *k*.

Hipótesis 2 (Asincronismo parcial)

Existe un atraso máximo de "d" iteraciones entre la iteración en que se utiliza una

variable y la iteración que la actualizó.

Denotamos como $\mathbf{x}^i(k)$ al vector $\mathbf{x}$ disponible en el procesador i en el momento de la iteración k, esto es:

$$\mathbf{x}^i(k) = \begin{bmatrix} \mathbf{x}_1^i(k) \\ \vdots \\ \mathbf{x}_p^i(k) \end{bmatrix}, \qquad \forall\, i \in \{1,\ldots,p\} \tag{2.21}$$

Usando esta notación podemos escribir el algoritmo asíncrono basado en (2.15) en la forma:

$$\mathbf{x}_i(k+1) = \mathbf{G}_i(\mathbf{x}^i(k)), \qquad \forall\, i \in \{1,\ldots,p\} \tag{2.22}$$

Esto es, cada procesador i actualiza $\mathbf{x}_i$ utilizando el valor disponible más actualizado de $\mathbf{x}$ al momento de iniciar la siguiente iteración.

Hipótesis 3 (Bloque-Lipschitz)
Cada operador $\mathbf{G}_i(\mathbf{x})$ *de* (*2.14*) *es Bloque-Lipschitz continuo en D, esto es:*

$$\left\| \mathbf{G}_i(\mathbf{x}) - \mathbf{G}_i(\mathbf{y}) \right\| \le \sum_{j=1}^{p} l_{ij} \left\| \mathbf{x}_j - \mathbf{y}_j \right\|, \qquad \forall\, \mathbf{x},\mathbf{y} \in D, \quad \text{con} \quad l_{ij} \ge 0 \tag{2.23}$$

Con el objetivo de derivar una condición suficiente de convergencia para el algoritmo asíncrono (2.22), en [4] se define el vector error $\mathbf{e} = [\mathbf{e}_1^T,\ldots,\mathbf{e}_p^T]^T \in \mathbb{R}^n$, $\mathbf{e}_i \in \mathbb{R}^{n_i}$ como:

$$\forall\, i,j \in \{1,..,p\}, \qquad \begin{bmatrix} \mathbf{e}_i(k+1) = \mathbf{x}_i(k+1) - \mathbf{x}_i^* \\ \mathbf{e}_j^i(k) = \mathbf{x}_j^i(k) - \mathbf{x}_j^* \end{bmatrix} \tag{2.24}$$

y el vector error reducido $\mathbf{z} = \left[\mathbf{z}_i, \ldots, \mathbf{z}_p\right]^T \in \Re^p$, $\mathbf{z}_i \in \Re$ como:

$$\forall i, j \in \{1,..,p\}, \qquad \begin{cases} \mathbf{z}_i(k+1) = \left\| \mathbf{e}_i(k+1) \right\| \\ \mathbf{z}^i_j(k) = \left\| \mathbf{e}^i_j(k) \right\| \end{cases} \tag{2.25}$$

Substrayendo (2.18) de (2.22) obtenemos $\forall i, j \in \{1,..,p\}$:

$$\mathbf{e}_i(k+1) = \mathbf{x}_i(k+1) - \mathbf{x}^*_i = \mathbf{G}_i(\mathbf{x}^i(k)) - \mathbf{G}_i(\mathbf{x}^*) \tag{2.26}$$

Tomando normas y usando la hipótesis 3:

$$\left\| \mathbf{e}_i(k+1) \right\| = \left\| \mathbf{G}_i(\mathbf{x}^i(k)) - \mathbf{G}_i(\mathbf{x}^*) \right\| \leq \sum_{j=1}^{p} l_{ij} \left\| \mathbf{x}^i_j(k) - \mathbf{x}^*_j \right\| \tag{2.27}$$

así:

$$\left\| \mathbf{e}_i(k+1) \right\| \leq \sum_{j=1}^{p} l_{ij} \left\| \mathbf{e}^i_j(k) \right\|, \qquad \forall i \in \{1,\ldots,p\} \tag{2.28}$$

que según (2.25) puede ser escrito como:

$$\mathbf{z}_i(k+1) \leq \sum_{j=1}^{p} l_{ij}\, \mathbf{z}^i_j(k), \qquad \forall i \in \{1,\ldots p\} \tag{2.29}$$

Para demostrar la convergencia del algoritmo asíncrono (2.22), es suficiente demostrar que el vector error reducido **z** que satisface la desigualdad (2.29) tiende a cero cuando $k \to \infty$. Para esto, usaremos el siguiente lema dado en [7]:

Lema 1 (Lema asíncrono de comparación)

Bajo la hipótesis 1 y 2, dada una matriz no negativa $\mathbf{H} = (h_{ij}) \geq 0$ *y un vector variante en el tiempo y no negativo* $\mathbf{z} \geq 0$*, satisfaciendo la inecuación*

$$\mathbf{z}_i(k+1) \leq \sum_{j=1}^{p} h_{ij}\, \mathbf{z}_j(d_j^i(k)), \qquad \forall i \in \{1,\ldots p\} \tag{2.30}$$

entonces, $\rho(\mathbf{H}) < 1$ *es una condición suficiente para que* $\mathbf{z}(k)$ *tienda exponencialmente a cero cuando* $k \to \infty$*, esto es*

si $\rho(\mathbf{H}) < 1$ entonces $\lim_{k\to\infty} \mathbf{z}(k) = \mathbf{0}$,

donde $\rho(\mathbf{H})$ *denota el radio espectral de la matriz* $\mathbf{H} = \{h_{ij}\} \in \Re^{p\times p}$.

El resultado inmediato de la aplicación del Lema 1 en la ecuación (2.29), tiene como consecuencia el teorema [4] que establece una condición suficiente de convergencia para el algoritmo asíncrono dado por (2.22).

Teorema 1 (Condición suficiente de convergencia)
Bajo las hipótesis 1 a 3, el algoritmo asíncrono representado por

$$\mathbf{x}_i(k+1) = \mathbf{G}_i(\mathbf{x}^i(k)), \qquad \forall i \in \{1,\ldots,p\}$$

converge a un único punto fijo $\mathbf{x}^*$ *en D si*

$$\rho(\mathbf{H}) < 1$$

donde $\mathbf{H}$ *está dada por*

$$\mathbf{H} = (h_{ij}) \in \Re^{pxp}, \qquad \text{con } h_{ij} = l_{ij}$$

NOTA 2: *La matriz de Comparación* $\mathbf{H}$ *es un límite superior (upper bound) de la función iteración* $\mathbf{H}(\mathbf{x}^k, k)$ [14].

En resumen, el radio espectral de la matriz de comparación es un parámetro que nos permitiría asegurar a priori que el algoritmo converge, y por consiguiente podría ser utilizado para escoger buenas particiones.

2.4 Método de Newton - Raphson

El método de Newton - Raphson, también conocido como método de Newton, es quizás el método más famoso para encontrar raíces de funciones no lineales. Este método requiere la evaluación de la misma función $f(x)$ así como de su derivada $f'(x)$, en un punto arbitrario x. El método de Newton-Raphson consiste geométricamente en extender la línea tangente por el punto actual $x(k)$ hasta cruzar el cero, luego fijar el siguiente punto $x(k+1)$ en la abscisa donde la tangente cruza el cero (ver figura 2.2).

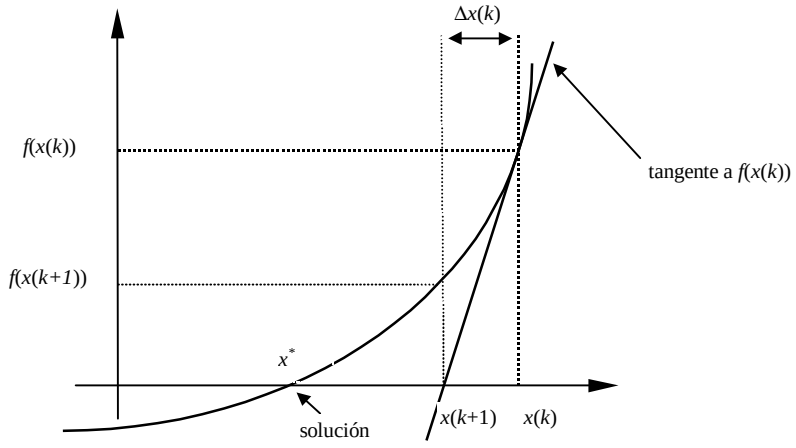


Figura 2.2: Método de Newton-Raphson

Algebraicamente, el método se deriva de la familiar expansión en serie de Taylor de una función en la vecindad de un punto,

$$f(x+\varepsilon) \approx f(x) + f'(x)\varepsilon + \frac{f''(x)}{2}\varepsilon^2 + \cdots$$

Para valores suficientemente pequeños de ε, y para funciones bien comportadas,

los términos no lineales pueden ser despreciados, así $f(x+_{o}) = 0$ implica

$$_{o} = - \frac{f(x)}{f'(x)}$$

Con esto, el algoritmo resultante es:

$$x(k+1) = x(k) - \frac{f(x(k))}{f'(x(k))}$$

El método de Newton-Raphson no está restringido a una sola dimensión, sino que fácilmente se puede generalizar a múltiples dimensiones [18]. La característica que hace al método poderoso y atractivo es su tasa de convergencia, ya que el mismo converge en forma cuadrática. Esta propiedad de fuerte convergencia hace que el método de Newton-Raphson sea apropiado para funciones cuya derivada sea continua, no nula en la vecindad de una raíz y pueda ser evaluada eficientemente.

Para un sistema unidimensional del tipo

$$f(x) = 0, \qquad f: \Re \to \Re, \qquad x, 0 \in \Re \tag{2.31}$$

el método de Newton-Raphson consiste en [16]:

1. Inicializar las iteraciones haciendo k = 0 y escoger una estimativa inicial para k = 0 que se denota $x(0)$.

2. Evaluar la función $f(x)$ en el punto $x = x(k)$

3. Comparar el valor calculado $f(x(k))$ con la tolerancia especificada ε; si se verifica que $|f(x(k))| \leq \varepsilon$, entonces la solución buscada será $x = x(k)$ por encontrarse dentro de los límites establecidos de tolerancia; si $|f(x(k))| > \varepsilon$, debe realizarse otra iteración.

4. Linealizar la función $f(x)$ en torno del punto $(\ x(k); f(x(k))\ )$ por medio de la serie de Taylor (ver figura 2.2):

$$f(\ x(k) + \underbrace{\Delta}\ x(k)\ ) \cong f(\ x(k)\ ) + f\,'(\ x(k)\ )\underbrace{\Delta}\ x(k) \qquad (2.32)$$

siendo $f\,'(x) = \dfrac{df}{dx}$

5. Resolver el problema linealizado, es decir, encontrar Δx tal que:

$$f(x(k)) + f\,'(x(k))\Delta\, x(k) = 0 \qquad (2.33)$$

El nuevo valor calculado de x será entonces

$$x(k+1) = x(k) + \underbrace{\Delta}\ x(k) \qquad (2.34)$$

donde

$$\underbrace{\Delta}\ x(k) = -\frac{f(x(k))}{f\,'(x(k))} \qquad (2.35)$$

6. Hacer k=k+1 y regresar al paso 2.

Generalizando la ecuación (2.31) conforme a la notación de (2.7) a (2.11) para el caso multivariable, se tiene el siguiente sistema no lineal de ecuaciones:

$$\Phi(\mathbf{x}) = \mathbf{F}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_n(\mathbf{x}) \end{bmatrix} = \mathbf{0}, \qquad \mathbf{F}: \Re^n \to \Re^n, \qquad \mathbf{x},\ \mathbf{0} \in \Re^n \qquad (2.36)$$

Conforme a lo descripto más arriba, la resolución de este sistema de ecuaciones consistirá en:

1. Hacer $k = 0$ y escoger $\mathbf{x}(0)$
2. Evaluar $\mathbf{F}(\mathbf{x}(k))$
3. Verificar si $|\mathbf{F}(\mathbf{x}(\mathrm{k}))| \leq \varepsilon$
4. Linealizar $\mathbf{F}(\mathbf{x})$ en torno al punto $(\mathbf{x}(k);\ (\mathbf{F}(\mathbf{x}(k)))$ según

$$\mathbf{F}(\mathbf{x}(k) + \Delta\mathbf{x}(k)) \cong \mathbf{F}(\mathbf{x}(k)) + \mathbf{J}(k)\Delta\mathbf{x}(k) \qquad (2.37)$$

5. Actualizar **x** según

$$\mathbf{x}(k+1) = \mathbf{x}(k) + \Delta\mathbf{x}(k) = \mathbf{x}(k) - \mathbf{J}^{-1}(k)\mathbf{F}(\mathbf{x}(k)) \qquad (2.38)$$

donde

$$\mathbf{J} = \left\{ \frac{\partial f_{\mathrm{i}}}{\partial x_{\mathrm{j}}} \right\}$$

Cuando se aplica el método a la resolución de un conjunto de *n* ecuaciones con *n* incógnitas, el jacobiano **J** es una matriz cuadrada cuyos elementos son iguales a las derivadas parciales de primer orden de cada función f_i con respecto a cada uno de los elementos del vector **x**. Para el sistema indicado en (2.36), se tendría entonces que:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \qquad (2.39)$$

En consecuencia, si queremos resolver (2.36) utilizando el método de Newton-

Raphson en un sistema de *p* procesadores, debemos:

1º) Descomponer $\mathbf{F}(\mathbf{x})$ en *p* funciones $\mathbf{F}_i(\mathbf{x})$ conforme (2.11), objetivo de este trabajo.

2º) Resolver $\mathbf{F}_i(\mathbf{x}) = \mathbf{0}$ en el procesador *i* conforme

$$\mathbf{x}_i(k+1) = \mathbf{x}_i(k) - \mathbf{J}_i^{-1}(k)\,\mathbf{F}_i(\mathbf{x}^i(k)) \qquad (2.40)$$

donde, por ejemplo, para *i*=1

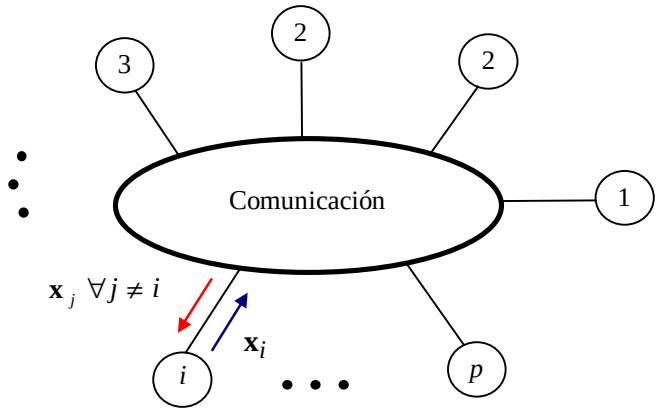
$$\mathbf{F}_1(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_{n_1}(\mathbf{x}) \end{bmatrix} \quad \text{y} \quad \mathbf{J}_1 = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_{n_1}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_{n_1}}{\partial x_1} & \cdots & \frac{\partial f_{n_1}}{\partial x_{n_1}} \end{bmatrix}$$

3º) Comunicar el valor de $\mathbf{x}_i$ a los demás procesadores que lo necesiten.

4º) Recibir valores de $\mathbf{x}_j$ enviados por los demás procesadores.

5º) Verificar si se llegó a la solución. De no ser así ir al 2° paso.

En la figura 2.3 podemos observar como cada procesador aplica el método de Newton-Raphson para actualizar su incógnita local $\mathbf{x}_i$ utilizando los resultados actualizados por los otros procesadores del sistema, previamente comunicados por éstos. Este proceso continúa en forma iterativa hasta que todo el conjunto llegue a la solución global del problema.



$$\mathbf{x}_i(k+1) = \mathbf{x}_i(k) - \mathbf{J}_i^{-1}(k)\Phi_i(\mathbf{x})$$

Figura 2.3 : Método de Newton-Raphson implementado en un sistema de *p* procesadores.

2.5 Solapamiento parcial

Las características particulares del sistema de ecuaciones a ser descompuesto pueden ocasionar que para ciertas ecuaciones (llamadas en el marco de este trabajo ecuaciones “criticas”) sea difícil realizar una decisión acertada en lo que respecta al procesador al cual deberán ser asignadas.

Una manera de salvar esta dificultad es asignar dicha ecuación a más de un procesador, realizando el llamado *solapamiento parcial* [10], buscando de esta manera mejorar la convergencia del método de resolución. Para explicar mejor el concepto de solapamiento parcial, se presenta a continuación un ejemplo ilustrativo:

- Ejemplo 2.5.1

Resolver el siguiente sistema lineal de ecuaciones, utilizando el método bloque-Jacobi en un sistema distribuido de dos procesadores.

$$\begin{bmatrix} 2 & 12 & 0 \\ 0.08 & 1 & 0.1 \\ 0 & 12 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{2.41}$$

Este sistema de ecuaciones puede ser representado por el siguiente grafo:

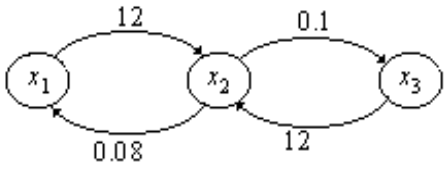


Figura 2.4: Grafo del sistema ejemplo

donde los nodos representan las incógnitas y las ramas representan los acoplamientos entre las incógnitas, obtenidos a partir de los coeficientes del sistema de ecuaciones. De este modo, la incógnita x_2 se encuentra muy fuertemente acoplada con las incógnitas x_1 y x_3, las cuales no se encuentran acopladas entre sí. La influencia de estos acoplamientos en el comportamiento del método iterativo se verificará en los algoritmos de actualización de incógnitas que serán presentados en este ejemplo.

En principio, el sistema puede ser descompuesto de tres formas posibles:

$\{x_1, x_2\}$ y $\{x_3\}$ (descomposición A)

$\{x_2, x_3\}$ y $\{x_1\}$ (descomposición B)

$\{x_1, x_3\}$ y $\{x_2\}$ (descomposición C)

De manera a considerar el criterio de convergencia presentado en la sección 2.3 (teorema 1), se calculará el radio espectral de la matriz de comparación para cada alternativa. Primero consideremos la descomposición A en la que a un procesador le son asignadas las dos primeras ecuaciones del sistema, mientras que al otro procesador se le asigna la tercera ecuación.

Así, y de acuerdo a lo mostrado en la sección 2.1, el algoritmo iterativo síncrono será:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 2 & 12 \\ 0.08 & 1 \end{bmatrix}^{-1} \left\{ \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} - \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} \left[x_3(k) \right] \right\} \qquad \text{para el procesador 1} \qquad (2.42.a)$$

$$\left[x_3(k+1) \right] = 3^{-1} \left\{ b_3 - \begin{bmatrix} 0 & 12 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} \right\} \qquad \text{para el procesador 2} \qquad (2.43.a)$$

Se puede apreciar que el elemento de valor 12 en la ecuación (2.43) refleja un alto grado de dependencia (acoplamiento) de la incógnita x_3 con respecto a la incógnita x_2, actualizada por el otro procesador.

Los elementos de la matriz de comparación **H** son las normas de las matrices bloques que componen la matriz de iteraciones. Entonces:

$$\mathbf{H} = \begin{bmatrix} 0 & \left\| \begin{bmatrix} 2 & 12 \\ 0.08 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} \right\| \\ \left\| 3^{-1} \begin{bmatrix} 0 & 12 \end{bmatrix} \right\| & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1.1539 \\ 4 & 0 \end{bmatrix} \qquad (2.44.a)$$

Dado que el radio espectral ρ de una matriz se define como el máximo de sus autovalores, para este ejemplo se tiene:

$$\rho(\mathbf{H}) = \sqrt{1.1539 \times 4} = 2.1484 > 1 \qquad (2.45.a)$$

Podemos decir así que la partición utilizada no presenta buenas perspectivas de convergencia, por no satisfacer la condición establecida en el Teorema 1.

Ahora, hagamos el mismo análisis para la segunda forma de partir el sistema (descomposición B). Así tenemos:

$$\begin{bmatrix} x_2(k+1) \\ x_3(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0.1 \\ 12 & 3 \end{bmatrix}^{-1} \left\{ \begin{bmatrix} b_2 \\ b3 \end{bmatrix} - \begin{bmatrix} 0.08 \\ 0 \end{bmatrix} \begin{bmatrix} x_1(k) \end{bmatrix} \right\} \qquad \text{para el procesador 1} \quad (2.42.b)$$

$$\begin{bmatrix} x_1(k+1) \end{bmatrix} = 2^{-1} \left\{ b_3 - \begin{bmatrix} 12 & 0 \end{bmatrix} \begin{bmatrix} x_2(k) \\ x_3(k) \end{bmatrix} \right\} \qquad \text{para el procesador 2} \quad (2.43.b)$$

La matriz de comparación correspondiente es:

$$\mathbf{H} = \begin{bmatrix} 0 & \left\| \begin{bmatrix} 1 & 0.1 \\ 12 & 3 \end{bmatrix}^{-1} \begin{bmatrix} 0.08 \\ 0 \end{bmatrix} \right\| \\ \left\| [2]^{-1} \begin{bmatrix} 12 & 0 \end{bmatrix} \right\| & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0.5333 \\ 6 & 0 \end{bmatrix} \qquad (2.44.b)$$

cuyo radio espectral es:

$$\rho_b(\mathbf{H}) = \sqrt{0.5333 \times 6} = 1.7888 > 1 \qquad (2.45.b)$$

Por último, consideremos la descomposición C. Así tenemos:

$$\begin{bmatrix} x_1(k+1) \\ x_3(k+1) \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}^{-1} \left\{ \begin{bmatrix} b_1 \\ b3 \end{bmatrix} - \begin{bmatrix} 12 \\ 12 \end{bmatrix} \begin{bmatrix} x_2(k) \end{bmatrix} \right\} \qquad \text{para el procesador 1} \quad (2.42.c)$$

$$\begin{bmatrix} x_2(k+1) \end{bmatrix} = 1^{-1} \left\{ b_2 - \begin{bmatrix} 0.08 & 0.1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_3(k) \end{bmatrix} \right\} \qquad \text{para el procesador 2} \quad (2.43.c)$$

La matriz de comparación correspondiente es:

$$\mathbf{H} = \begin{bmatrix} 0 & \left\| \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}^{-1} \begin{bmatrix} 12 \\ 12 \end{bmatrix} \right\| \\ \left\| [1]^{-1} \begin{bmatrix} 0.08 & 0.1 \end{bmatrix} \right\| & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0.18 \\ 6 & 0 \end{bmatrix} \qquad (2.44.c)$$

cuyo radio espectral es:

$$\rho(\mathbf{H}) = \sqrt{0.18 \times 6} = 1.0392 > 1 \qquad (2.45.c)$$

Se puede observar que ninguna de las tres descomposiciones cumple con la condición suficiente de convergencia, esto es $\rho(\mathbf{H}) < 1$. En consecuencia, utilizaremos la técnica de solapamiento parcial para intentar descomponer el problema (2.41).

Para esto, se partirá el problema anterior de la siguiente manera: la segunda ecuación del sistema (ecuación crítica) se replicará en ambos procesadores, esto es, el procesador 1 resolverá las ecuaciones 1 y 2 del sistema, mientras el procesador 2 resolverá las ecuaciones 2 y 3. En este caso, el nuevo sistema de ecuaciones a ser resuelto, denominado *sistema expandido* [10], tiene una dimensión expandida *n'*= 4, y puede ser representado por:

$$\begin{bmatrix} 2 & 12 & 0 & 0 \\ 0.08 & 1 & 0 & 0.1 \\ 0.08 & 0 & 1 & 0.1 \\ 0 & 0 & 12 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_{21} \\ x_{22} \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_2 \\ b_3 \end{bmatrix} \qquad (2.46)$$

y en forma de grafo por la figura 2.5.

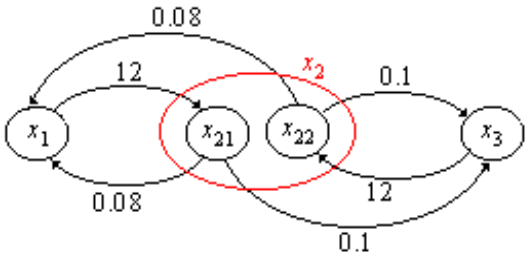


Figura 2.5: Grafo del sistema ejemplo expandido

Note que ahora tenemos 2 versiones diferentes de la variable x_2, estas son: x_{21} y x_{22}. En consecuencia, cada procesador implementará el algoritmo de Jacobi según se muestra

a continuación:

$$\begin{bmatrix} x_1(k+1) \\ x_{21}(k+1) \end{bmatrix} = \begin{bmatrix} 2 & 12 \\ 0.08 & 1 \end{bmatrix}^{-1} \left\{ \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ 0 & 0.1 \end{bmatrix} \begin{bmatrix} x_{22}(k) \\ x_3(k) \end{bmatrix} \right\} \quad \text{para el procesador 1} \qquad (2.47)$$

$$\begin{bmatrix} x_{22}(k+1) \\ x_3(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0.1 \\ 12 & 3 \end{bmatrix}^{-1} \left\{ \begin{bmatrix} b_2 \\ b_3 \end{bmatrix} - \begin{bmatrix} 0.08 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_{21}(k) \end{bmatrix} \right\} \quad \text{para el procesador 2} \qquad (2.48)$$

En forma análoga al caso anterior, se construye la matriz **H** de comparación:

$$\mathbf{H} = \begin{bmatrix} 0 & \left\| \begin{bmatrix} 2 & 12 \\ 0.08 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 & 0 \\ 0 & 0.1 \end{bmatrix} \right\| \\ \left\| \begin{bmatrix} 1 & 0.1 \\ 12 & 3 \end{bmatrix}^{-1} \begin{bmatrix} 0.08 & 0 \\ 0 & 0 \end{bmatrix} \right\| & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1.1539 \\ 0.5333 & 0 \end{bmatrix} \qquad (2.49)$$

cuyo radio espectral $_b$ (**H**) será:

$$_b(\mathbf{H}) = \sqrt{1.1539 \times 0.5333} = 0.7845 < 1$$

que por el Teorema 1 tiene asegurada su convergencia, aún en un contexto computacional asíncrono.

En la tabla 2.1, se muestran los resultados experimentales obtenidos al resolver el problema ejemplo utilizando un sistema distribuido compuesto por dos procesadores de igual performance. Se puede observar que solo en el caso de solapamiento parcial el sistema de ecuaciones converge a la una solución.

	Descomposición A	Descomposición B	Descomposición C	Con Solapamiento
$b_{\acute{o}}$ (**H**)	2.1484	1.7888	1.0392	0.7845
número de iteraciones	no converge	no converge	no converge	29
tiempo	no converge	no converge	no converge	1 s.

Tabla 2.1 : Resultados de la resolución del sistema ejemplo.

En conclusión, el solapamiento parcial permite resolver en forma alternativa problemas que no podrían ser resueltos con una simple partición de las ecuaciones correspondientes a dicho problema.

CAPITULO 3: METODOS DE DESCOMPOSICION EXISTENTES

Hoy en día se dispone de varios métodos para la descomposición de sistemas de ecuaciones de gran porte (ver capítulo 1), presentando algunos de ellos características que los hacen muy atractivos para el procesamiento paralelo.

El presente capítulo considera primero la *búsqueda exhaustiva* que como se verá, no es un método práctico para hallar la mejor descomposición. Se expondrán luego los principios básicos de los dos métodos de descomposición más utilizados hasta ahora, que han sido implementados con éxito en diversas aplicaciones de ingeniería y la ciencia en general. Dichos métodos son la *Descomposición* ε [24 - 26] y el *Método de la Semilla* [29, 30]. Este último método es de especial importancia en el marco del presente trabajo, por servir de base a la presente propuesta.

3.1 Búsqueda exhaustiva

Siempre que se desee conocer la mejor descomposición de entre un número dado de posibilidades, conforme a un criterio dado, se puede hacer una *búsqueda exhaustiva*, analizando todas las posibles combinaciones hasta encontrar la descomposición óptima.

Considerando la conocida formula de combinaciones $C_m^n = \frac{n!}{m!(n-m)!}$, se puede conocer el número de casos posibles (*ncp*) que tiene que ser considerado. Usando la nomenclatura del capítulo 2, se tiene:

para $p = 1$, $ncp = C_n^n = 1$

para $p = 2$, $ncp = C_{n_1}^n = C_{n_2}^n$, donde $n = n_1 + n_2$

para $p = 3$, $ncp = C_{n_1}^{n}\ C_{n_2}^{n\ \ n_1}$, donde $n = n_1 + n_2 + n_3$

para $p = 4$, $ncp = C_{n_1}^{n}\ C_{n_2}^{n\ \ n_1} C_{n_3}^{n\ \ (n_1\ ,\ n_2)}$, donde $n = n_1 + n_2 + n_3 + n_4$

Generalizando para p procesadores se tiene:

$$ncp = C_{n_1}^{n}\ C_{n_2}^{n\ \ n_1}\ C_{n_3}^{n\ \ (n_1\ ,\ n_2)}\ \cdots\ C_{n_p\ 1}^{n\ \ (n_1\ ,\ n_2\ ,\ \cdots\ ,\ n_p\ 2)}$$

donde $n = n_1 + n_2 + n_3 + \cdots + n_p$

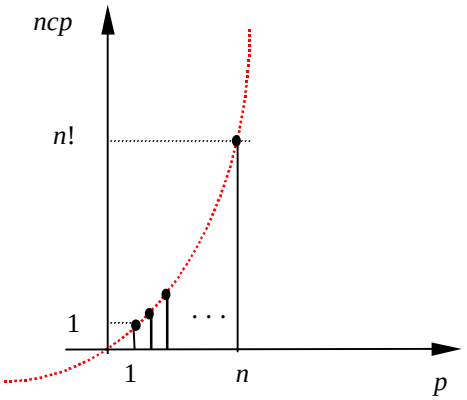


Figura 3.1 : Gráfico del número de casos posibles (*ncp*) de la búsqueda exhaustiva.

Ejemplo: Sea un sistema de $n = 100$ ecuaciones. Si tenemos dos procesadores y necesitamos descomponer de forma tal que $n_1 = 60$ y $n_2 = 40$, entonces

$$ncp = C_{60}^{100} = \frac{100!}{60!(100\_60)!} = \frac{100!}{60!\,40!} = 1.37 \times 10^{28}$$

Si tenemos tres procesadores y necesitamos descomponer de forma tal que $n_1 = 60$, $n_2 = 20$ y $n_3 = 20$, entonces

$$ncp = C_{60}^{100}\ C_{20}^{100\ \ 60} = \frac{100!}{60!(100\_60)!}\ \frac{40!}{20!(40\_20)!} = \frac{100!}{60!\,20!\,20!} = 1.89 \times 10^{39}$$

Como se puede observar, el número de posibilidades (que crece en forma factorial) es tan grande (ver figura 3.1), que es impracticable analizar cada una de ellas, teniendo en cuenta las dimensiones de los sistemas de ecuaciones hoy utilizados en problemas aplicativos. De aquí la importancia de disponer de un método práctico para encontrar una buena descomposición, aunque ésta no sea la óptima.

3.2 La Descomposición ε

Desde finales de la década del 60, con la publicación del trabajo de Carré [9], se puso de manifiesto que el acoplamiento entre los subsistemas en que el sistema de ecuaciones es descompuesto influye de manera decisiva en la eficiencia de los métodos de resolución, aún cuando originalmente dichos métodos hayan sido aplicados en un ambiente secuencial y no paralelo.

Basándose en este hecho, la *Descomposición* $_{\varepsilon}$ [24 - 26] introduce un método de gran simplicidad. Lo primero que se hace es asociar el sistema de ecuaciones a un grafo donde los nodos son las incógnitas y las ramas o arcos son los coeficientes del sistema de ecuaciones. Una vez realizado ésto, el objetivo básico de este método es separar el grafo en los puntos en los que el acoplamiento (o dependencia) entre incógnitas sea lo más débil posible, a través de la eliminación de las ramas cuyo valor de acoplamiento sea menor que un límite inferior ε. Esta eliminación de ramas se repite utilizando valores de ε cada vez mayores, hasta que el sistema de ecuaciones $\Phi(\mathbf{x})$ quede efectivamente descompuesto en varios subsistemas $\Phi_i(\mathbf{x})$. Como se verá más adelante, las dimensiones de los subsistemas no pueden ser controladas de una manera efectiva, al depender éstas exclusivamente de las características particulares del sistema de ecuaciones.

Para ilustrar la aplicación de la *Descomposición* $_{\varepsilon}$, considérese el sistema de ecuaciones

$$\mathbf{Ax} = \mathbf{b}$$

donde

$$\mathbf{A} = \begin{bmatrix} 100 & 1.7 & 1.9 & 0.4 & 0 & 0 & 0 & 0 \\ 1.7 & 12 & 2 & 0 & 0 & 0 & 0 & 0 \\ 1.9 & 2 & 132 & 0.1 & 0 & 0.15 & 0 & 1 \\ 0.4 & 0 & 0.1 & 25 & 1.9 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 1.9 & 80 & 1.1 & 0 & 0 \\ 0 & 0 & 0.15 & 0.5 & 1.1 & 100 & 2.2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2.2 & 12 & 1.5 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1.5 & 50 \end{bmatrix} \quad y \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} ,$$

representado por el grafo que se muestra en la figura 3.2, donde los arcos no son direccionados porque la matriz es simétrica.

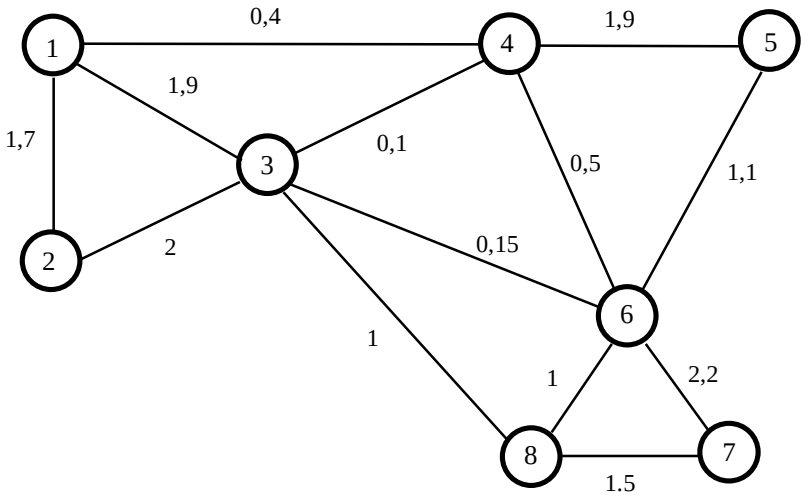


Figura 3.2 : Descomposición ε: sistema de ecuaciones ejemplo.

Los nodos del grafo representan las incógnitas del sistema de ecuaciones, mientras que las ramas representan las acoplamientos que existen entre las mismas. Los números adosados a las ramas representan los valores de dichos acoplamientos.

A continuación, se determina un límite inferior ε que será utilizado como

referencia para seleccionar las ramas que serán eliminadas. Para el ejemplo, se utilizará inicialmente ε = 0.6 , eliminándose del grafo las ramas cuyos valores de acoplamiento sean menores que 0.6. Así, el sistema quedará como se muestra en la figura 3.3.

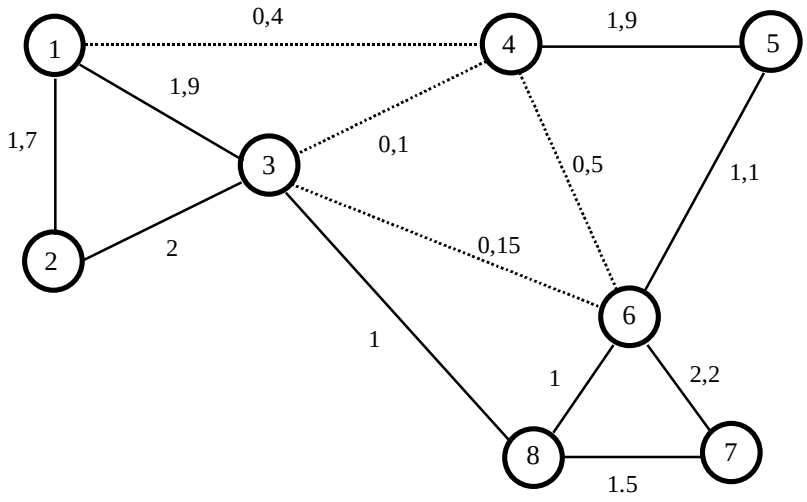


Figura 3.3 : Aplicación de la Descomposición ε: situación 1 (ε = 0.6).

Como aún no se ha logrado particionar el sistema, el paso anterior se repite tomando un valor de ε mayor, por ejemplo ε = 1.2. Eliminando las ramas con valores menores que 1.2 , el sistema quedará separado en 3 subsistemas menores según se observa en la figura 3.4.

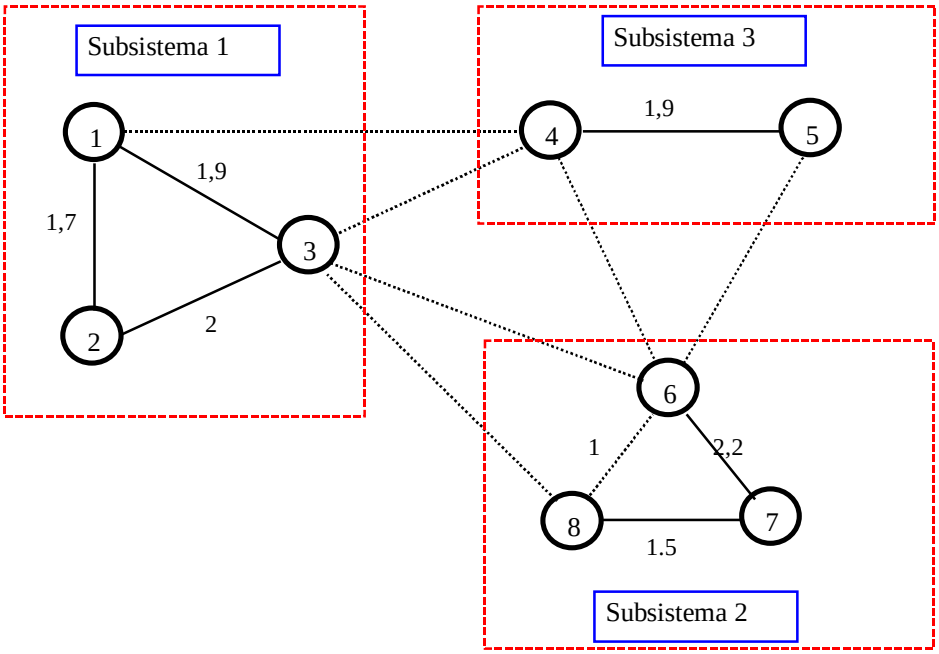


Figura 3.4 : Aplicación de la Descomposición ε: situación 2 (ε = 1.2).

Puede verse que los subsistemas generados por el método expuesto están débilmente acoplados, que era lo que el método buscaba, pero las dimensiones relativas y el número de subsistemas obtenidos dependieron exclusivamente de las características particulares del sistema de ecuaciones.

3.3 El Método de la Semilla

Este método presentado por M.H.Vale, D.M. Falcão y E. Kaszkurewicz [29], fue concebido exclusivamente para ser aplicado a sistemas eléctricos y es una metodología de partición basada en la identificación de grupos de incógnitas fuertemente acopladas, para descomponer así el sistema dado en subsistemas que contengan a cada uno de dichos grupos.

Una vez asociado el sistema de ecuaciones a un grafo donde los nodos son las

incógnitas y las ramas o lados son los coeficientes, el principio básico de esta metodología es la formación de subsistemas a partir de un dado conjunto de incógnitas *semillas.* Los subsistemas se forman agregando incógnitas una a una a estas semillas, utilizándose como criterio de agrupación un *ranking* de pesos previamente asignados a todas las incógnitas del sistema. En cada paso del proceso, la siguiente incógnita a ser agregada a un subsistema en formación es aquella incógnita adyacente (unida por alguna rama al subsistema) de mayor peso y que aún no haya sido asociada por algún otro subsistema en formación.

El ranking de pesos P_i de las incógnitas se realiza en base a la fórmula

$$P_i = \sum_{l \in \mathcal{R}_i} B_l^{B_l / M}, \qquad \text{con} \qquad M = \sum_{l \in \mathcal{R}_T} \frac{B_l}{b} \tag{3.1}$$

donde:

$\mathcal{R}_i$: Conjunto de todas las ramas conectadas a la incógnita *i*.

$\mathcal{R}_T$: Conjunto de todas las ramas del sistema eléctrico.

B_l: Susceptancia de la rama *l*. (acoplamiento)

b: Número de ramas del sistema eléctrico.

La selección de las incógnitas semillas se realiza en base a la determinación de un conjunto de incógnitas candidatas a semillas. Este conjunto contiene a todas las incógnitas que poseen un valor de peso superior a un valor pre-determinado *vlim*.

A partir de cada incógnita del conjunto así determinado, e independientemente de las otras incógnitas del conjunto, se agrupan las incógnitas más pesadas de su entorno, pretendiéndose con esto identificar a las incógnitas que se rodean con acoplamientos más fuertes. No es preciso agrupar a todas las incógnitas del sistema en torno a cada incógnita candidata, sino sólo a *ngrup* incógnitas.

Una vez realizado esto, se calculan los valores de las sumatorias de los pesos de las

incógnitas agrupadas alrededor de cada candidata a semilla, seleccionándose las incógnitas semillas de entre aquellas que posean los mayores valores de sumatoria. Para evitar que sean seleccionadas incógnitas muy próximas, con fuertes acoplamientos entre sí, se impone que una incógnita semilla no se encuentre entre las *nvec* primeras incógnitas del agrupamiento de las incógnitas semillas seleccionadas anteriormente.

A continuación, y a manera de ilustración, se aplicará el método de la semilla en la descomposición del sistema ejemplo utilizado en la sección 3.2, buscando descomponer la sistema de ecuaciones para resolverlo en dos procesadores.

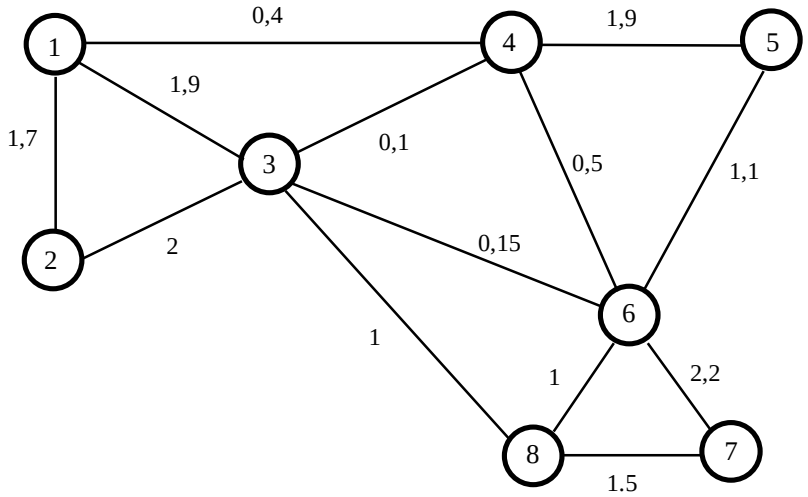


Figura 3.5 : Método de la semilla: sistema de ecuaciones ejemplo

Los valores de los parámetros a ser utilizados serán:

- *vlim* = 5.5
- *ngrup* = 2
- *nvec* = 2

Estos valores fueron determinados de manera a ofrecer una explicación lo más clara posible del método aquí expuesto. Para problemas reales, dichos valores son establecidos conforme a la experiencia y conocimiento que el operador posea del sistema eléctrico analizado [29].

Como primer paso, se calculan los pesos de las incógnitas conforme a (3.1) y se los ordena en forma decreciente, resultando así:

$P_3 = 8.6$
$P_6 = 7.91$
$P_7 = 5.96$
$P_1 = 5.65$
$P_2 = 5.34$
$P_4 = 5.08$
$P_5 = 3.88$
$P_8 = 3.66$

Tomando *vlim*=5.5, se determina el conjunto de incógnitas candidatas a semillas, cuyos elementos serán las incógnitas x_3, x_6, x_7 y x_1, así ordenadas por peso (ver tabla 3.1). A continuación, alrededor de estas 4 incógnitas se agrupan las *ngrup* incógnitas más pesadas, calculándose luego la sumatoria de los pesos de las incógnitas agrupadas. Para la selección realizada con *ngrup*=2, se tienen los siguientes agrupamientos:

Candidata	Incógnitas agrupadas	$\sum P_i$
x_3	x_6, x_7	22.47
x_6	x_3, x_7	22.47
x_7	x_6, x_3	22.47
x_1	x_3, x_6	22.16

Tabla 3.1: Agrupamientos de incógnitas

A pesar de existir coincidencia en las sumatorias de pesos, la primera incógnita seleccionada como semilla es la incógnita x_3, por poseer ésta individualmente el mayor valor de peso. La segunda incógnita con el mayor valor de sumatoria es la incógnita x_6, pero antes de seleccionarla como semilla debe verificarse que no se encuentre entre las *nvec* primeras incógnitas agrupadas alrededor de la incógnita x_3. Tomando *nvec*=2, se puede ver que la incógnita x_6 no puede ser seleccionada como semilla, por ser la primera incógnita del agrupamiento de la primera semilla seleccionada. La incógnita x_7, que es la

siguiente candidata a semilla con la mayor sumatoria de pesos, tampoco puede ser seleccionada, por encontrarse entre las *nvec* primeras incógnitas del agrupamiento de la incógnita x_3. Se puede ver que la incógnita x_1, que es la última incógnita candidata, no se encuentra siquiera en el agrupamiento de la incógnita x_3, por lo que se la selecciona como segunda semilla.

De este modo, las incógnitas x_1 y x_3 quedan seleccionadas como semillas y ya se dispone de la cantidad de semillas necesarias de forma a descomponer el sistema en dos subsistemas, uno para cada procesador. Es de esperar que las semillas seleccionadas sean centros de agrupamientos de incógnitas, de manera a dar origen a subsistemas poco acoplados entre sí.

Una vez determinadas las incógnitas semillas, se inicia el proceso de descomposición. Para esto, cada semilla selecciona a la más pesada de entre sus adyacentes y la incluye en su subsistema en formación. Tenemos así que, para nuestro ejemplo, la semilla x_3 incluirá a la incógnita x_6, mientras que la semilla x_1 hace lo propio con la incógnita x_2. Los subsistemas en formación quedarán entonces como se muestra en la figura 3.6.

El proceso continúa hasta que no hayan más incógnitas por ser anexadas, quedando así el sistema de ecuaciones descompuesto en tantos subsistemas como semillas haya. Para nuestro ejemplo, al finalizar el proceso de partición, el sistema quedará descompuesto como se muestra en la figura 3.7, donde se puede ver que el método de la semilla solamente genera particiones balanceadas, es decir, constituidas por subproblemas de dimensiones similares.

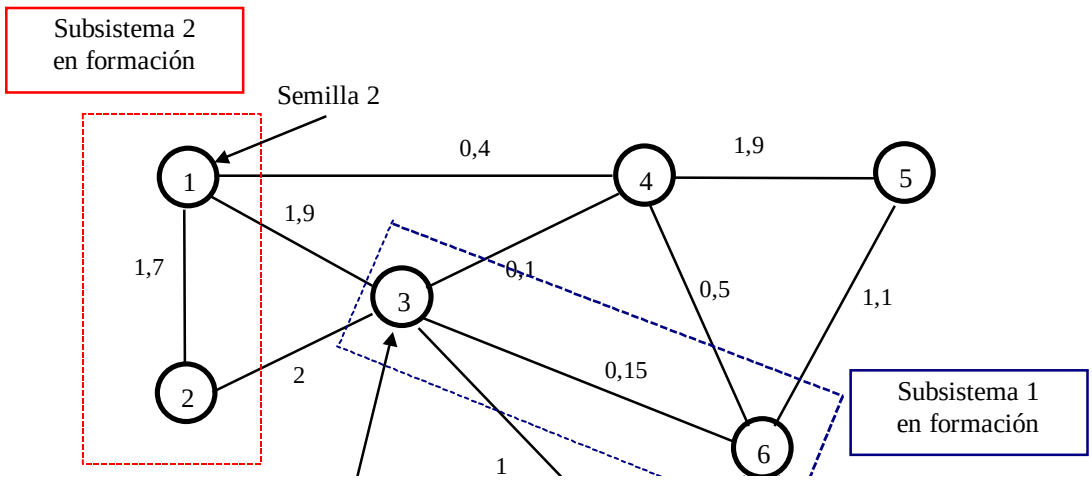


Figura 3.6 : Aplicación del método de la semilla.

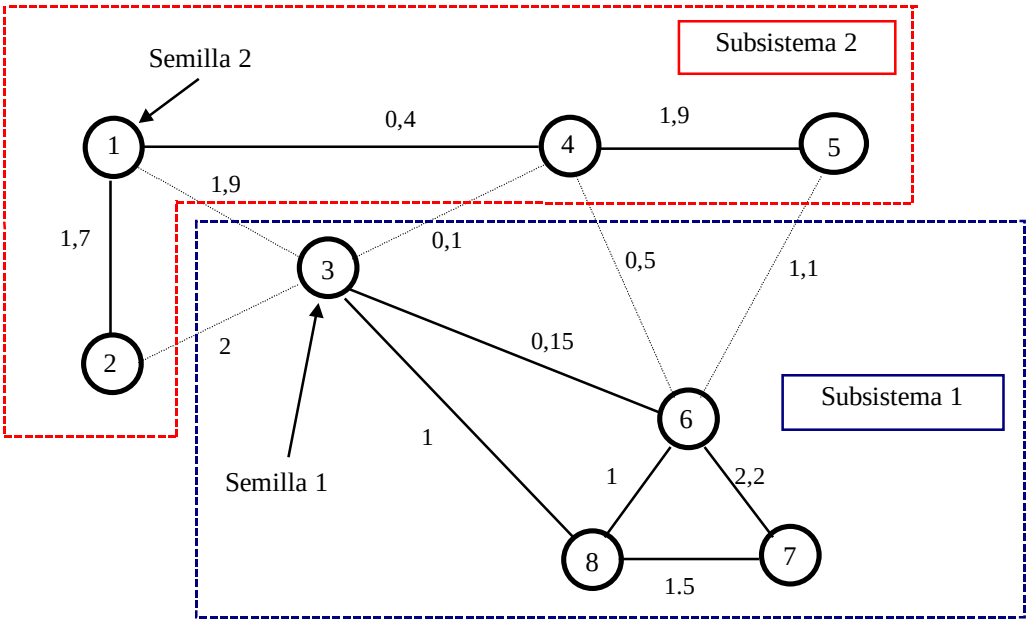


Figura 3.7 : Método de la semilla: partición generada

De este modo, el presente método genera particiones constituidas por subsistemas poco acoplados entre sí, lo que resulta muy beneficioso para el desempeño de los métodos de resolución bloque-iterativos. Con todo, el método no posee dominio sobre las dimensiones relativas de los subsistemas generados, ni considera la posibilidad de realizar solapamiento parcial.

3.4 Comparación entre los Métodos de Descomposición

Ya en las secciones anteriores se mencionaron las ventajas y desventajas de cada uno de los métodos existentes. A continuación, se resumen éstos en la tabla 3.2.

	Ventajas	Desventajas
Búsqueda Exhaustiva	• Se obtiene la partición óptima.	• El número de combinaciones posibles crece en forma factorial, por lo que es impracticable.
Descomposición ε.	• Simple. • Fácil de implementar. • Utilizado actualmente. • Sugiere buenas particiones.	• No se tiene control sobre el número de subsistemas. • No se tiene control sobre el tamaño de cada subsistema.
Método de la Semilla	• Tiene en cuenta los centros de acoplamiento de incógnitas. • Funciona bien para sistemas computacionales homogéneos. • La cantidad de particiones candidatas crece en forma lineal con el tamaño del problema.	• Solo para sistemas eléctricos (matriz simétrica). • No contempla sistemas distribuidos heterogéneos. • Dificultades para seleccionar la mejor partición.

Tabla 3.2: Comparación de Métodos de Partición

El objetivo del presente trabajo es proponer un método de descomposición de un sistema de ecuaciones en varios subsistemas menores, de forma a facilitar su resolución en un sistema distribuido heterogéneo. En este sentido, se puede observar que el *Método de la Semilla* tiene varias características que lo hacen muy atractivo para este objetivo, como ser:

- considerar los centros de acoplamiento de incógnitas;
- la cantidad de particiones a evaluar crece en forma lineal (y no factorial) con el tamaño del problema [30];
- control sobre el número de subsistemas.

Entonces, basado en el *Método de la Semilla*, el presente trabajo propone un

método que salva sus limitaciones (ver tabla 3.2), permitiendo así una mejor distribución de la carga computacional asignada a cada procesador, a la vez de ofrecer la posibilidad de realizar solapamiento parcial con miras a mejorar la convergencia del método de resolución empleado, conforme se explica en el próximo capítulo.

CAPITULO 4: METODO DE DESCOMPOSICION PROPUESTO

4.1 Consideraciones Iniciales

Una buena descomposición de un sistema de ecuaciones es sumamente importante, dado que sin ella no sería posible obtener resultados satisfactorios al utilizar los métodos bloque-iterativos en la resolución paralela/distribuida de los diversos problemas de ingeniería.

Se pudo observar que la manera en la cual un sistema es descompuesto tiene un efecto significativo en la convergencia de los métodos bloque-iterativos. Para obtener buenos resultados en términos de convergencia, se debe descomponer el sistema de tal forma que el acoplamiento entre subsistemas sea lo más débil posible, donde el acoplamiento estará definido en función al método de resolución a ser empleado.

Con respecto al balanceamiento de carga computacional, se ha podido constatar que los métodos de partición descriptos en el Capítulo 3 no se ocupan de manera especial en generar una descomposición acorde con las capacidades relativas de procesamiento de las máquinas que constituyen el sistema distribuido, esto es, consideran sistemas distribuidos homogéneos, constituidos por procesadores de idéntico desempeño.

Esto último es de singular importancia porque, especialmente en nuestro medio, es sumamente difícil conseguir varios ordenadores de características similares. Es más común el disponer de una red compuesta por varias máquinas de capacidad y desempeño diferentes, constituyendo así un ambiente computacional heterogéneo, dado que los procesadores de la red tienen capacidades de procesamiento bastante variadas.

Considerando la posibilidad de operar en un ambiente de dichas características, la descomposición también estará condicionada por la capacidad de procesamiento relativo de cada máquina del sistema distribuido. Este factor determina las dimensiones de los

subproblemas asignados a cada procesador.

De acuerdo a lo anterior, la técnica de descomposición debe perseguir los siguientes objetivos de interés:

- Convergencia del método de solución: en este sentido, debe buscarse tanto un acoplamiento débil como un número reducido de acoplamientos entre los subsistemas interconectados; esto último es con el fin de minimizar la comunicación entre procesadores y mejorar la convergencia del conjunto.
- Eficiencia computacional del método de solución en términos del procesamiento paralelo: debe buscarse para esto el balanceamiento de la carga computacional entre los distintos procesadores que componen el sistema distribuido. El parámetro a ser utilizado como criterio de evaluación de la eficiencia computacional es el tiempo total de ejecución del método de solución en la resolución del problema dado.

Una vez identificados los objetivos perseguidos, la formulación del problema de la descomposición podría entenderse como: "*una técnica que debe descomponer un sistema de ecuaciones en un número dado de subsistemas interconectados entre sí de tal forma que su resolución en un sistema distribuido heterogéneo utilizando métodos bloque-iterativos sea obtenida en el menor tiempo de procesamiento posible*".

Atendiendo a este objetivo, el método de descomposición propuesto utiliza una matriz cuadrada no negativa $\mathbf{M} = \{m_{ij}\}$ ($m_{ij} \geq 0$) de dimensión $n_x n$ y elementos diagonales no nulos ($m_{ii} \neq 0$), cuyos elementos m_{ij} y m_{ji} representan el acoplamiento existente entre las incógnitas x_i y x_j. La matriz **M** puede por ejemplo, ser la matriz de coeficientes del sistema de ecuaciones o una versión derivada de la misma. Por ejemplo en el caso lineal, (**Ax** = **b**), los elementos m_{ij} de la matriz **M** están dados por $m_{ij} = | a_{ij} |$, donde **A** es la matriz de coeficientes de sistema. Si el sistema a ser resuelto fuera no lineal, la matriz **M** a ser utilizada dependerá del método de resolución. En los estudios experimentales

incluidos en este trabajo se resolvió el problema del Flujo de Potencia de un sistema eléctrico, para lo cual se tomó como matriz **M** a una matriz derivada de la matriz de admitancias **Y** de la red eléctrica en estudio (ver el Apéndice B).

Decimos que las incógnitas x_i y x_j no son adyacentes si $m_{ij} = m_{ji} = 0$; caso contrario x_i y x_j son adyacentes. Si m_{ij} y m_{ji} son pequeños (en el caso límite $m_{ij} \rightarrow 0$ y $m_{ji} \rightarrow 0$) el acoplamiento existente entre las incógnitas x_i y x_j es débil, con poca dependencia entre ellas, pudiendo por lo tanto cada una de ellas formar parte de subsistemas diferentes. En consecuencia, el método buscará separar el sistema de ecuaciones en los puntos donde el acoplamiento entre las incógnitas sea lo más débil posible, identificando para eso a los *centros de agrupamiento* de incógnitas.

Suponiendo que el sistema de ecuaciones será resuelto utilizando un sistema distribuido de *p* procesadores, el principio básico del método consistirá en la formación de los *p* subsistemas a partir de *p* incógnitas iniciales, llamadas *semillas* las cuales son identificadas como centros de acoplamiento de incógnitas.

Una vez determinadas estas semillas para cada subsistema, de acuerdo al algoritmo de selección de semillas que será presentado en esta sección, es necesario adicionar las demás incógnitas al subsistema adecuado hasta tener el sistema de ecuaciones totalmente particionado. Para realizar esto, se asocia la incógnita adyacente de mayor peso a cada subsistema en formación. El proceso de agrupamiento de incógnitas continua hasta que todas ellas hayan sido asignadas a algún subsistema. En caso de obtenerse más de una descomposición para un mismo sistema, un criterio de selección de descomposiciones será presentado con vista a elegir la partición más conveniente.

Basado en el principio descrito, se propone un método compuesto por cuatro etapas, esquematizado en a la figura 4.1.

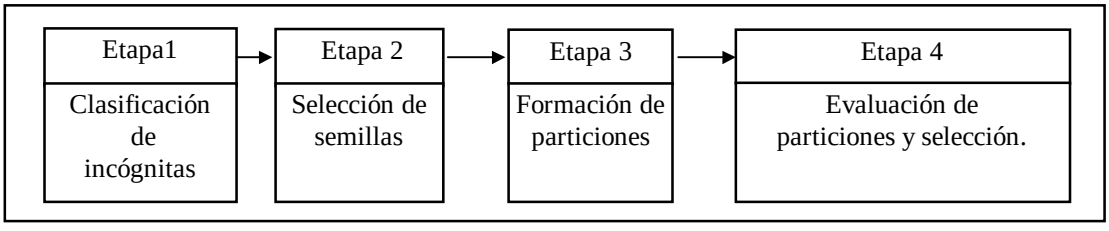


Figura 4.1 : Etapas del método de partición propuesto.

4.2 Etapa 1: Clasificación de incógnitas

Como se ha resaltado con anterioridad, uno de los objetivos principales a ser perseguido por un buen método de partición es separar el sistemas de ecuaciones de modo que el acoplamiento entre incógnitas en subsistemas diferentes sea lo mas débil posible. Para esto, se impone conocer que tan acoplada se encuentra una incógnita con las demás incógnitas del sistema de ecuaciones. Este grado de acoplamiento será medido en el marco de este trabajo por el valor de un cierto "peso" calculado para cada incógnita.

Sea $nc = número\ de\ m_{ij} \neq 0$

Se denota como D a la media aritmética de los elementos no diagonales de **M** excluyendo a los elementos nulos; esto es:

$$D = \frac{1}{nc} \sum_{i=1}^{n} \sum_{j=1,\, j \neq i}^{n} m_{ij} \neq 0 \qquad (4.1)$$

y sea

$$m'_{ij} = (\text{máximo } \{ m_{ij} ; m_{ji} \}) / m_{ii} \neq 0 \qquad (4.2)$$

el máximo entre m_{ij} y m_{ji} normalizado con respecto a m_{ii} , y

$$z_{ij} = m'_{ij} / D \neq 0 \qquad (4.3)$$

el valor normalizado de m'_{ij} respecto a la media *D*.

Se define el peso P_i de una incógnita x_i como:

$$P_i = \sum_{j=1,\, j \neq i}^{n} (m'_{ij})^{z_{ij}} \qquad \forall i \in \{1,\ldots,n\},\ \forall m'_{ij} \neq 0 \qquad (4.4)$$

La clasificación de las incógnitas es realizada calculando el peso de cada incógnita y estableciendo un *ranking* de las mismas conforme a sus pesos P_i, según el algoritmo mostrado a continuación:

Entrada: matriz **M**
DESDE *i* = 1 hasta *n* /* para cada una de las *n* incógnitas. */ Calcular P_i según la ecuación (4.4); Incluir a P_i en el ranking ordenado de incógnitas;
Salida: Pesos P_i de las *n* incógnitas ordenadas según un ranking de pesos

Figura 4.2 : Algoritmo de clasificación de incógnitas.

Es importante observar que el valor del peso P_i busca representar si la incógnita en cuestión posee acoplamientos fuertes (valor elevado de m'_{ij}) o acoplamientos débiles (valor pequeño de m'_{ij}) con las otras incógnitas de la red. El exponente z_{ij} aparece en la expresión para que las ramas con elevados valores de m'_{ij} contribuyan más significativamente al valor del peso, buscando diferenciar incógnitas con varios acoplamientos con bajos valores de m'_{ij} de otras incógnitas con pocos acoplamientos pero con elevados valores de m'_{ij} .

Nótese que otras definiciones de pesos pueden ser dadas, como por ejemplo:

a) $P_i = \prod_{j=1, j \neq i}^{n} (m_{ij})^{\frac{m_{ij}}{D}}$ $\forall i \in \{1,\ldots,n\},\ \forall m_{ij} \neq 0$

b) $P_i = \prod_{j=1, j \neq i}^{n} (\frac{m_{ij}}{m_{ii}})^{\frac{m_{ij}}{m_{ii} D}}$ $\forall i \in \{1,\ldots,n\},\ \forall m_{ij} \neq 0$

c) $P_i = \prod_{j=1, j \neq i}^{n} (\frac{m_{ij} + m_{ji}}{2})^{\frac{m_{ij} + m_{ji}}{2 D}}$ $\forall i \in \{1,\ldots,n\},\ \forall m_{ij} \neq 0$

d) $P_i = \prod_{j=1, j \neq i}^{n} (\frac{m_{ij} + m_{ji}}{2 m_{ii}})^{\frac{m_{ij} + m_{ji}}{2 m_{ii} D}}$ $\forall i \in \{1,\ldots,n\},\ \forall m_{ij} \neq 0$

e) $P_i = \prod_{j=1, j \neq i}^{n} (m_{ij})^{\frac{m_{ij}}{D}} + \prod_{j=1, j \neq i}^{n} (m_{ji})^{\frac{m_{ji}}{D}}$ $\forall i \in \{1,\ldots,n\},\ \forall m_{ij} \neq 0$

f) $P_i = \prod_{j=1, j \neq i}^{n} (\frac{m_{ij}}{m_{ii}})^{\frac{m_{ij}}{m_{ii} D}} + \prod_{j=1, j \neq i}^{n} (\frac{m_{ji}}{m_{ii}})^{\frac{m_{ji}}{m_{ii} D}}$ $\forall i \in \{1,\ldots,n\},\ \forall m_{ij} \neq 0$

La definición de peso utilizada en el marco del presente trabajo (ecuación 4.4) es la que resultó experimentalmente más conveniente en las implementaciones realizadas.

4.3 Etapa 2: Selección de semillas

La determinación de las incógnitas que harán el papel de semillas representa una importante etapa en el método propuesto. Además de determinar el número de subsistemas en que se irá a descomponer el sistema de ecuaciones y dependiendo del conjunto de semillas determinado, pueden ser generadas diferentes descomposiciones. Consecuentemente, se obtendrán diferentes resultados en términos de eficiencia computacional del método de solución.

Se verificó en los resultados experimentales que los agrupamientos de incógnitas

fuertemente acopladas (elevado valor de m'_{ij}) no deben ser separados en diferentes subsistemas, con vistas a atender los objetivos anteriormente citados. Así, se optó por seleccionar como semillas aquellas incógnitas que sean centro de agrupamiento de incógnitas y que no están fuertemente acopladas entre sí conforme al algoritmo ilustrado en la figura 4.3.

Para entender dicho algoritmo de selección de semillas, es necesario definir los parámetros: *vlim*, *ngrup* y *nvec*, así como los conjuntos ***K***, ***S***, ***I*** y ***CIA***.

- Valor límite: *vlim*

  Como las semillas deben ser centros de aglutinamiento de incógnitas, es de esperar que las de mayor peso (las más fuertemente acopladas al resto de las incógnitas) sean las mejores candidatas a semilla. Para esto, se define un parámetro *vlim* tal que toda incógnita x_i que satisfaga $P_i \geq vlim$ sea una candidata a semilla. En principio, todas las incógnitas podrían ser evaluadas como candidatas a semillas haciendo *vlim* = 0, pero esto implicaría un alto costo computacional a causa de la gran cantidad de particiones que tendrían que ser evaluadas en la etapa 4. Como consecuencia de esto, se adoptan como *vlim* valores que limiten el número de candidatas a semillas a una cantidad razonable [29, 30]. En el estudio de los sistemas eléctricos paradigmas de la IEEE, se observó una cierta tendencia al agrupamiento en los valores de los pesos de las incógnitas; es decir, existen por lo general grupos de incógnitas que poseen pesos de valores aproximadamente iguales. Este hecho se constituye en una referencia interesante a la hora de decidir sobre el valor del parámetro *vlim*.

- Número de agrupamiento: *ngrup*

  ngrup es un parámetro que indica cuantas incógnitas deben ser agrupadas alrededor de cada candidata a semilla para luego, realizando una sumatoria de sus pesos, decidir cuales serán las *p* semillas a ser utilizadas.

- Radio mínimo de vecindad: *nvec*

  nvec es un parámetro que se utiliza para evitar que incógnitas fuertemente acopladas entre sí sean semillas al mismo tiempo.

El valor de estos tres parámetros puede ser determinado por el usuario según el conocimiento que tenga del sistema en estudio. En todo caso, pueden ser utilizados criterios estadísticos para determinar estos parámetros. Por ejemplo, Vale et al.[29, 30] recomienda los siguientes valores:

vlim: dos valores de peso tales que un cierto número de los pesos calculados se encuentren por encima de ellos.

ngrup: 1%,2%,5%,10% y 15% de la relación *n/p* entre el número de incógnitas *n* del sistema y el número de procesadores *p*.

nvec: 1% y 10% de *n*/*p*.

Con respecto a los estudios experimentales realizados aplicando el algoritmo de selección de semillas, se puede mencionar que dependiendo de la dimensión del sistema a ser descompuesto y de la capacidad del procesador en el cual el método de es implementado, se tomaron con éxito valores de *vlim* tales que sean consideradas como candidatas a semillas hasta 11 incógnitas de entre 117 posibles, con valores de *ngrup* y *nvec* del orden del 8% de la relación *n/p*.

- Conjunto ***K***:

  Es el conjunto de incógnitas candidatas a ser semillas; sus elementos son las incógnitas x_i con un peso $P_i \geq vlim$.

- Conjunto ***S***:

  Es el conjunto de incógnitas que serán semillas de una partición. Al iniciar el algoritmo de selección de semillas el conjunto ***S*** está vacío y al finalizar contiene las *p*

incógnitas semillas. Si se desea, se podrán tener varios conjuntos ***S***, escogiendo diferentes ternas *vlim*, *ngrup* y *nvec*. Debe tenerse en cuenta, sin embargo, que dos o más ternas pueden generar la misma partición.

- Conjunto ***I***:

  Por cada incógnita x_i candidata a ser semilla existe un conjunto $\boldsymbol{I}_i$, que tiene como primer elemento a la misma incógnita candidata, a la cual se irán anexando *ngrup* incógnitas que definirán su "región de influencia". Esta región de influencia define al conjunto de incógnitas que de forma directa o indirecta (a través de otras incógnitas) están fuertemente acopladas entre sí.

- Conjunto ***CIA***:

  Es el **C**onjunto de **I**ncógnitas **A**dyacentes a las incógnitas del conjunto $\boldsymbol{I_i}$, e indica cuáles incógnitas están disponibles para ser incluidas en este último conjunto.

El principio básico del método de selección de semillas es el siguiente:

1º) Se determina el conjunto de incógnitas que serían buenas candidatas a semillas (conjunto ***K***) individualizando a las incógnitas cuyo valor de peso sea igual o mayor a *vlim*.

2º) A partir de cada una de las incógnitas x_i del conjunto ***K***, e independientemente de las otras incógnitas de este conjunto, se asocian a ella sucesivamente (se incluyen en su conjunto $\boldsymbol{I_i}$) las incógnitas más pesadas de entre las adyacentes a la agrupación (pertenecientes al conjunto $\boldsymbol{CIA_i}$). A medida que tal agrupamiento es hecho, se calcula la sumatoria de los pesos de todas las incógnitas agrupadas en $\boldsymbol{I_i}$, pretendiéndose con esto identificar las incógnitas de ***K*** que se rodean con las mayores ligaciones. La agrupación continúa hasta que la incógnita candidata haya agrupado *ngrup* incógnitas.

3º) Una vez calculados los valores de las sumatorias relativas a cada incógnita de ***K***, las incógnitas semillas son escogidas de entre aquellas que poseen los mayores valores de

sumatoria. Para evitar que sean escogidas incógnitas muy próximas, con fuertes ligaciones entre ellas, se impone que una incógnita semilla no se encuentre entre las *nvec* primeras incógnitas asociadas en el agrupamiento de las semillas previamente seleccionadas.

El pseudocódigo del algoritmo implementado para seleccionar las semillas se muestra en la figura 4.3.

Entradas: terna *vlim*, *ngrup* y *nvec*, matriz **M**, pesos de las incógnitas P_i , nº de procesadores *p*.
vlim, es el peso mínimo requerido para considerar una incógnita como candidata a semilla;
ngrup, es el nº de incógnitas a agrupar alrededor de cada candidata a semilla y
nvec, es el parámetro utilizado para evitar que dos incógnitas muy acopladas sean semillas.

```
Inicializar el conjunto K como vacío;                    /* Conjunto de incógnitas candidatas a semillas */
PARA cada una de las incógnitas x_i
    SI peso P_i ≥ vlim ENTONCES
        Incluir la incógnita x_i en K ;
PARA cada una de las incógnitas x_i en K
    Inicializar el conjunto I_i como vacío;              /* Conjunto de incógnitas agrupadas a las */
                                                         /* candidatas a semillas                  */
    Incluir la incógnita x_i en I_i ;
    Inicializar el conjunto CIA_i como vacío;   /* Conjunto de Incógnitas Adyacentes al conjunto I_i */
    Incluir en CIA_i las incógnitas adyacentes a la incógnita x_i ;
    DESDE 1 hasta ngrup          /* agrupar ngrup incógnitas alrededor de cada candidata */
        Incluir la incógnita de mayor peso del CIA_i al I_i ;
        Eliminar esta incógnita del CIA_i ,
        Incluir en CIA_i , las nuevas incógnitas adyacentes a la recientemente movida;
PARA cada conjunto I_i
    Calcular la sumatoria de los pesos de todas sus incógnitas en I_i ;
Inicializar el conjunto S;                           /* S: Conjunto de semillas de una partición */
Seleccionar de K la incógnita x_k cuyo conjunto I posea mayor sumatoria de peso ;
Incluir x_k en S como primera semilla ;
Eliminar x_k de K ;
MIENTRAS el número de semillas ≤ p
    Seleccionar en K la incógnita x_s que posea la mayor sumatoria de pesos ;
    SI x_s no está entre las nvec primeras incógnitas de los conjuntos I de las semillas
    ya seleccionadas ENTONCES
        Elegir x_s como semilla ;
        Incluir x_s en S ;
        Eliminar x_s de K;
    DE_LO_CONTRARIO
        Eliminar x_s de K ;
```

Salida: Para cada terna se tendrá un conjunto ***S*** de *p* semillas

Figura 4.3 : Algoritmo de selección de semillas.

4.4 Etapa 3: Generación de la partición

Una vez determinadas las semillas para cada subsistema de acuerdo al algoritmo de selección de semillas, es necesario adicionar las demás incógnitas a los subsistemas más adecuados. Cada una de los *p* subsistemas que formarán la partición tiene como incógnita inicial a su semilla, y a ella se irán anexando otras incógnitas, teniendo en cuenta para ello los pesos P_i , los valores de acoplamiento m'_{ij} y el hecho de optar por realizar solapamiento parcial o no, en caso de empates.

Un aspecto importante en la formación de la partición es la necesidad de asignar a cada procesador un subproblema de dimensión proporcional a su performance relativa con vistas a lograr el equilibrio de la carga computacional. Para ello, se definen a continuación los vectores **w**, **C** *y* **Q**

- *Performance Relativa entre Procesadores*: **w**

  Se define $\mathbf{w} \in \mathbb{N}^p$ como el vector de *Performance Relativa entre Procesadores*. Dicha performance relativa es la relación que existe entre las capacidades de procesamiento de los procesadores. Por ejemplo, $\mathbf{w} = [\,2 \quad 1\,]^T$ implica que el procesador 1 puede procesar el doble de ecuaciones que el procesador 2 en el mismo periodo de tiempo.
  En este trabajo se pretende que, dados el sistema de ecuaciones y un sistema distribuido con performance relativa **w** conocida, hallar una partición que resuelva el problema en estudio de forma eficiente.

- *Balanceamiento de carga*: **C**

  Se define al vector Balanceamiento de carga $\mathbf{C} \in \mathbb{N}^p$ como un vector de estado de la forma $\mathbf{C} = [\,c_1 \quad \cdots \quad c_p\,]^T$, donde los c_i representan la cantidad de incógnitas agrupadas en el subconjunto $\boldsymbol{J}_i$ en un momento dado. Al finalizar el proceso de

partición, el vector **C** indicará cuantas incógnitas serán asignadas a cada procesador. El método buscará que los vectores **w** y **C** sean paralelos; esto es, se encuentren en siguiente relación:

$$\mathbf{C} = \alpha\, \mathbf{w}$$

donde $\alpha \geq 0$ es una constante escalar.

- *Cupo Parcial*: **Q**

  Se define al vector Cupo Parcial $\mathbf{Q} \in \mathbb{N}^p$ como un vector de estado de la forma $\mathbf{Q} = [\, q_1 \ \cdots \ q_p \,]^T$, donde los q_i se obtienen de acuerdo a la relación $q_i = c_i/w_i$.

  Idealmente, al terminar la computación tendría que verificarse que

$$\| \mathbf{Q} \|_\infty = q_i = \alpha \qquad \forall\, i \in \{1,\ldots,p\}$$

(4.5)

aunque esto puede no darse por no ser los valores de las componentes del vector **w** divisores exactos del número de incógnitas del sistema a descomponer.

- *Límite de Solapamiento Parcial*: *LimOver*

  Se define *LimOver* como el mínimo valor requerido del peso de una incógnita para realizar solapamiento parcial. Si el peso P_i de un una incógnita es mayor o igual que *LimOver*, entonces se realiza el solapamiento parcial de esa incógnita en caso de empate. Este valor puede ser determinado empíricamente utilizando valores estadísticos. Por ejemplo, una opción apropiada para este parámetro sería tomar los valores de *vlim* que se han utilizado, al ser estos referencias válidas de la posición relativa de una incógnita con respecto a las demás en el ranking de pesos.

- *Conjunto **J***:

  Se define como el conjunto de incógnitas de un partición que serán asignadas a un procesador para ser resueltas. Existen *p* conjuntos ***J*** . Al iniciar la generación de la partición el conjunto $\boldsymbol{J}_i$ contiene solo la semilla y luego se irán anexando las demás incógnitas que formarán parte del subsistema.

En el proceso de generación de las particiones, cada semilla selecciona como candidata entre sus adyacentes a la incógnita que posea mayor peso y que a su vez no sea parte de otro subsistema. Si no existen coincidencias entre las candidatas de las distintas semillas, dichas incógnitas candidatas son asociadas a las semillas que la escogieron, constituyéndose así en subsistemas en formación. El proceso continúa de forma análoga: cada subsistema selecciona la incógnita más pesada y aun no agrupada de entre sus adyacentes. Si no existen coincidencias de candidatas, se procede a la inclusión de dichas candidatas en los subsistemas correspondientes. Con el fin de obtener el balanceo de carga deseado, solamente seleccionarán candidatas aquellos subsistemas que se encuentran en desventaja frente a los otros subsistemas. Con esto se busca que los subsistemas generados posean las dimensiones requeridas.

En el caso de existir coincidencias entre candidatas, se analiza el peso de la incógnita en disputa. Si el mismo es menor que un valor requerido (*LimOver*) para realizar solapamiento, se asigna la incógnita a aquel subsistema con el que tenga una ligación más fuerte. En caso que el peso de la incógnita sea mayor que el límite de solapamiento, se asigna dicha incógnita a todos los subsistemas que la reclamen como candidata.

En la figura 4.4 se presenta el algoritmo de formación de particiones.

Entradas: conjunto de semillas $\boldsymbol{S} = \{s_1,...,s_p\}$, matriz **M**, pesos de las incógnitas P_i , *LimOver*:
LimOver: mínimo peso requerido para realizar solapamiento parcial.

Inicializar los vectores $\mathbf{C} \in \mathbf{N}^p$ y $\mathbf{Q} \in \mathbf{N}^p$; /* vectores de estado para controlar el desbalanceo *w*; */
/* c_i es el nº de incógnitas en el subproblema *i* en */
/* un momento dado; $q_i = c_i/w_i$ */

PARA cada semilla $s_i \subset \boldsymbol{S}$
- Inicializar el conjunto $\boldsymbol{J_i}$ como vacío; /* conjunto de incógnitas de un subproblema */
- Incluir en $\boldsymbol{J_i}$ la semilla s_i ;
- Inicializar un conjunto $\boldsymbol{CIA_i}$ como vacío; /* ***C***onjunto de ***I***ncógnitas ***A***dyacentes */
- Incluir en $\boldsymbol{CIA_i}$ las incógnitas adyacentes a la semilla s_i ;

Actualizar los vectores **C** y **Q** ;

MIENTRAS existan incógnitas no agrupadas
- **PARA** todos los conjuntos $\boldsymbol{J}_i$ que necesitan anexar incógnitas
  - Seleccionar como incógnita candidata a ser incluida a la de mayor peso de su ***CIA*** ;
  - Ver si existen coincidencias de candidatas ;
- **SI** existe coincidencia en la incógnita candidata $\hat{x}_k$ **ENTONCES**
  - Seleccionar el valor del mayor acoplamiento entre las incógnitas ya incluidas en $\boldsymbol{J}_i$ y la incógnita candidata $\hat{x}_k$ a ser incluida en $\boldsymbol{J}_i$;
  - **SI** existen coincidencias de valor de mayor acoplamiento **ENTONCES**
    - **SI** se tiene opción de solapamiento **ENTONCES**
      - **SI** el peso de la incógnita $\hat{x}_k$ es $\geq$ *LimOver* **ENTONCES**
        - Incluir $\hat{x}_k$ en todos los subconjuntos ***J*** coincidentes ;
      - **DE_LO_CONTRARIO**
        - Incluir $\hat{x}_k$ en el primer ***J*** que pelea por $\hat{x}_k$;
    - **DE_LO_CONTRARIO**
      - Incluir $\hat{x}_k$ en el primer ***J*** que pelea por $\hat{x}_k$;
  - **DE_LO_CONTRARIO**
    - Incluir $\hat{x}_k$ en ***J*** con el que tenga mayor acoplamiento ;
- **DE_LO_CONTRARIO**
  - Incluir $\hat{x}_k$ en el primer $\boldsymbol{J}_i$;
- Eliminar $\hat{x}_k$ de todos los ***CIA***s de ***J*** ;
- Incluir en los ***CIA***s correspondientes las incógnitas adyacentes a las incluidas ;
- Actualizar **C** y **Q** ;

Salida: partición en *p* subsistemas.

Figura 4.4 : Algoritmo de formación de particiones.

4.5 Etapa 4: Evaluación de particiones y selección

Hay diversas formas de generar descomposiciones de un sistema de ecuaciones, sea en forma intuitiva o utilizando algún criterio automático. En esta sección se presentará un *criterio selectivo de descomposiciones*, que tiene como objetivo indicar, entre un conjunto dado de descomposiciones del sistema, aquellas que presenten las mejores características en términos del desempeño computacional de los métodos de resolución bloque-iterativos.

En efecto, si se utilizan distintas ternas de parámetros *vlim, ngrup* y *nvec* pueden ser determinados varios conjuntos diferentes de semillas que a su vez servirán de pie para generar diferentes particiones. Debido a la necesidad de poseer una referencia previa sobre la calidad de la partición de forma a evitar la necesidad de resolver cada sistema para conocer su desempeño, es sumamente útil disponer de un parámetro que refleje la calidad de una partición.

En [30] se proponen varios parámetros posibles de selección, pudiéndose citar entre estos a:

- El número total de acoplamientos que unen las distintos subsistemas en que el sistema de ecuaciones es descompuesto.
- El máximo de los números de acoplamientos entre dos subsistemas cualesquiera.
- La sumatoria total del módulo de las ramas (los coeficientes del sistema de ecuaciones) que son seccionadas por la partición.
- La mayor diferencia entre los números de ramas internas de dos subsistemas cualquiera.
- La menor sumatoria de valores absolutos de las ramas (los coeficientes del sistema de ecuaciones) internas a los subsistemas.

En el marco de dicho trabajo, fue sugerido como el mejor parámetro a la sumatoria

de todos los valores absolutos de las ligaciones entre incógnitas separadas por la partición, el cual llamaremos Parámetro de Acoplamiento “Par_A”; esto es,

Par_A = $\Sigma \mid m_{ij} \mid$ para todo par de incógnitas (x_i, x_j) que se encuentren en subsistemas diferentes. (4.6)

Conforme a lo expuesto en el Capítulo 2 sección 2.3, el radio espectral ρ (**H**) de la matriz de comparación sería un buen parámetro de selección por el hecho de proporcionar información sobre la convergencia del algoritmo de resolución, y en consecuencia, sobre el desempeño computacional de la partición analizada. Por lo tanto, en este trabajo se propone como parámetro de selección de descomposiciones al radio espectral de la matriz de comparación **H**, siendo elegida como óptima aquella partición que presente el menor valor de ρ (**H**) . La efectividad de este parámetro de selección se analizará en los estudios experimentales presentados en el Capítulo 7.

Conviene acotar aquí que el radio espectral de la matriz de comparación **H** es un límite superior (*upper bound*) del radio espectral de la matriz de iteración, que en el caso lineal es, rigurosamente hablando, el parámetro óptimo para indicar si el algoritmo de resolución convergerá a la solución o no (condición necesaria y suficiente de convergencia). Sin embargo, y a pesar de sus ventajas, el cálculo de este último parámetro reviste de una dificultad computacional mayor que la misma resolución del sistema de ecuaciones en su totalidad, por lo que no tiene sentido su utilización. En efecto, al tener la matriz de iteración dimensión nxn, el cálculo de su radio espectral es mucho más complejo que el cálculo del radio espectral de la matriz de comparación, cuya dimensión es sólo de pxp.

Para consolidar el método presentado, el siguiente capítulo presenta un ejemplo

ilustrativo.

Entrada: las diversas particiones generadas por el algoritmo de particiones

PARA cada partición
  Calcular $_b$ (**H**)
  Incluir el valor de $_b$ (**H**) en el ranking
Elegir como mejor partición la de menor valor de $_b$ (**H**).

Salidas: la mejor partición generada por el método propuesto, ranking de particiones.

Figura 4.5 : Algoritmo de selección de particiones.

CAPITULO 5: UN EJEMPLO ILUSTRATIVO

Se presenta en este capítulo una aplicación sencilla del método propuesto. El ejemplo a ser utilizado fue especialmente concebido para permitir una explicación bien práctica del proceso de partición propuesto en el Capítulo 4.

5.1 Presentación del problema

Descomponer el siguiente sistema lineal de ecuaciones de manera a resolverlo utilizando 2 procesadores con performance relativa $\mathbf{w} = [4\ 1]^T$

$$\mathbf{Ax} = \mathbf{b} \tag{5.1}$$

donde

$$\mathbf{A} = \begin{bmatrix} 100 & 0.2 & 0 & 1 & 0.5 & 0 & 1.2 & 2 & 1.5 & 2 \\ 0.2 & 12 & 0.1 & 0 & 3 & 0 & 1.1 & 1 & 1.2 & 0 \\ 0 & 0.1 & 132 & 1 & 0.1 & 1 & 0 & 0 & 2 & 0.1 \\ 1 & 0 & 1 & 25 & 0 & 1 & 2 & 1.1 & 1 & 0.2 \\ 0.5 & 3 & 0.1 & 0 & 80 & 1 & 0.1 & 0 & 2 & 2 \\ 0 & 0 & 1 & 1 & 1 & 100 & 0.1 & 0.1 & 0.5 & 2 \\ 1.2 & 1.1 & 0 & 2 & 0.1 & 0.1 & 12 & 0 & 0.1 & 0 \\ 2 & 1 & 0 & 1.1 & 0 & 0.1 & 0 & 50 & 0 & 2 \\ 1.5 & 1.2 & 2 & 1 & 2 & 0.5 & 0.1 & 0 & 76 & 0 \\ 2 & 0 & 0.1 & 0.2 & 2 & 2 & 0 & 2 & 0 & 99 \end{bmatrix} \quad \text{y} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \end{bmatrix}$$

De la sección 4.1, la matriz **M** a ser utilizada es idéntica a la matriz de coeficientes del sistema de ecuaciones **A**, es decir $m_{ij} = a_{ij}; \quad \forall i, j \in \{1,\ldots,10\}$.

5.2 Clasificación de las incógnitas

Para el sistema dado se calcularon los pesos de las 10 incógnitas conforme a la ecuación (4.4) (ver Tabla 5.1). Con base en los valores calculados, se puede realizar ahora

la clasificación de las incógnitas en función a sus pesos, de acuerdo al algoritmo de la figura 4.2. El resultado obtenido se ilustra en la figura 5.1.

Incógnita	Peso
x_6	6.76551
x_1	6.67490
x_5	6.63105
x_9	6.60877
x_4	6.21757
x_3	5.85905
x_{10}	5.69746
x_7	5.29716
x_2	5.06927
x_8	4.61528

Tabla 5.1: Pesos de las incógnitas

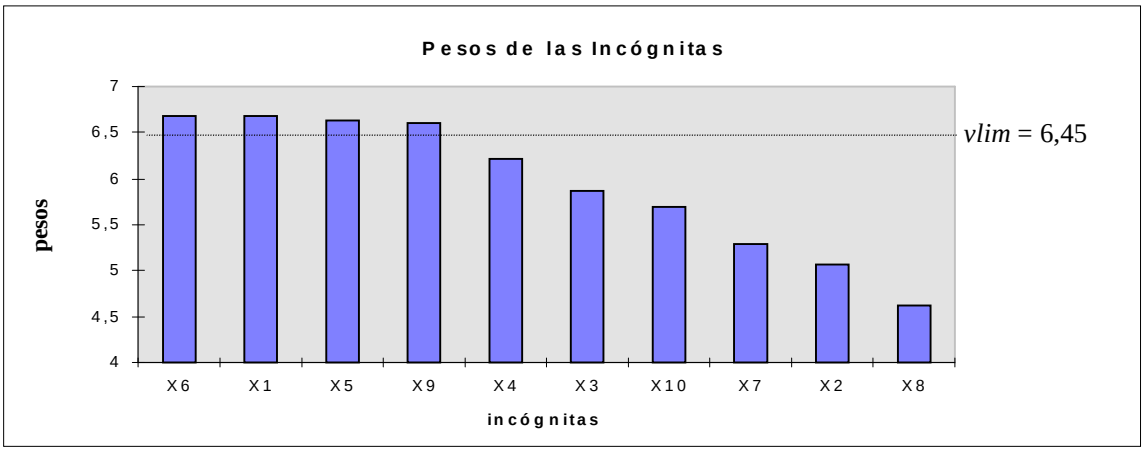


Figura 5.1 : Pesos de las incógnitas del sistema ejemplo.

5.3 Selección de semillas

Combinando distintos valores de *vlim*, *ngrup* y *nvec* podrán ser seleccionados varios conjuntos diferentes de semillas. A continuación se determinarán los valores a ser utilizados, siguiendo criterios que fueron encontrados válidos para facilitar el

entendimiento del lector. Otros criterios más prácticos ya fueron presentados en la sección 4.3.

- ***vlim***: Observando la figura 5.1, se identifican dos agrupaciones de peso, separadas por la ordenada 6.45. Por lo tanto, este último valor puede ser una opción válida para el parámetro *vlim*. Note que tomando valores menores, la gran mayoría de las incógnitas, si no la totalidad, serían seleccionadas como candidatas a semillas, lo que aumentaría de manera significativa el trabajo computacional a ser realizado. Por otro lado, si se toman valores de *vlim* muy elevados, se corre el riesgo de no disponer de suficientes incógnitas candidatas como para poder realizar una partición adecuada.
- ***ngrup*:** Debido a la pequeña dimensión del problema, no sería práctico agrupar alrededor de cada incógnita candidata a semilla un número muy elevado de barras, pues ello resultaría en agrupamientos semejantes, impidiendo así obtener el número requerido de semillas. Se agruparán en este caso solo 3 incógnitas además de la incógnita candidata, esto es, *ngrup=3*.
- ***nvec*:** Para evitar que dos semillas formen parte del mismo agrupamiento de incógnitas, el parámetro *nvec* deberá asegurar un distanciamiento razonable entre ellas. En este caso, y debido nuevamente a la pequeña dimensión del problema, *nvec* =2 proporcionaría un *radio de vecindad* mínimo aceptable.

Con base en el valor de *vlim* adoptado, se puede determinar ahora los elementos del conjunto ***K***, que contendrá a las incógnitas candidatas a semillas. Se incluirán en él a cada incógnita cuyo peso sea mayor o igual a 6.45. Tenemos así:

$$\boldsymbol{K} = \{ x_6, x_1, x_5, x_9 \}$$

A continuación se realizará el proceso de agrupamiento de incógnitas alrededor de cada una de las 4 candidatas. Para el efecto, la tabla 5.2 muestra como ejemplo el proceso para la incógnita x_6. La primera columna de dicha tabla (***I***) contendrá a las incógnitas que forman parte del agrupamiento, mientras que la segunda columna (***CIA***) está destinada a

contener a todas las incógnitas adyacentes al agrupamiento en formación. Se puede ver que inicialmente, en la primera columna solo se encuentra la incógnita candidata (en este caso x_6) y en la segunda sus incógnitas adyacentes.

$\boldsymbol{I_6}$	$\boldsymbol{CIA_6}$
$\boldsymbol{x_6}$	$x_5\ x_9\ x_4\ x_3\ x_{10}\ x_7\ x_8$

Tabla 5.2: Cuadro de agrupamiento de incógnitas: situación 1

Para continuar con el proceso, se selecciona seguidamente la incógnita de mayor peso de entre las adyacentes a x_6 y se la incluye entre las incógnitas del agrupamiento en formación. En consecuencia, la incógnita x_5 pasa a integrar la columna $\boldsymbol{I_6}$ conforme se muestra en la tabla 5.3. Nótese que en la columna ***CIA*** será eliminada x_5 por no encontrarse ya disponible para un futuro anexamiento, mientras que las incógnitas adyacentes a x_5 y que aún no se encontraban en ***CIA*** son incluidas ahora en dicha columna. La incógnita eliminada de ***CIA***, por haber pasado a la columna ***I***, está señalada por una barra cruzada ($\not{x}_5$) en la tabla 5.3.

$\boldsymbol{I_6}$	$\boldsymbol{CIA_6}$
$\boldsymbol{x_6}$	$\not{x}_5\ x_9\ x_4\ x_3\ x_{10}\ x_7\ x_8$
x_5	$x_1\ x_2$

Tabla 5.3 : Cuadro de agrupamiento de incógnitas: situación 2.

El proceso anterior continúa hasta que *ngrup* incógnitas hayan sido asociadas a la incógnita candidata a semilla en la columna ***I***. Así son incluidas la incógnitas x_1 y x_9, ya que *ngrup* = 3. La tabla quedará entonces como se muestra a continuación:

I_6	CIA_6
x_6	x_5 x_9 x_4 x_3 x_{10} x_7 x_8
x_5	x_1 x_2
x_1	
x_9	

Tabla 5.4 : Cuadro de agrupamiento de incógnitas.

Este mismo procedimiento se aplica a cada una de las incógnitas del conjunto ***K***.

Se muestra a continuación la tabla obtenida una vez finalizado este proceso. Se han agregado columnas donde se colocaron los pesos de las incógnitas agrupadas, así como casillas donde se encuentran las sumatorias de los pesos de las incógnitas agrupadas alrededor de cada candidata a semilla.

I_6	CIA_6	I_1	CIA_1
x_6	x_5 x_9 x_4 x_3 x_{10} x_7 x_8	x_1	x_5 x_9 x_4 x_{10} x_7 x_2 x_8
x_5	x_1 x_2	x_5	x_6 x_3
x_1		x_6	
x_9		x_9	
$\sum P_i$	26.68	$\sum P_i$	26.68

I_5	CIA_5	I_9	CIA_9
x_5	x_6 x_1 x_9 x_3 x_{10} x_7 x_2	x_9	x_6 x_1 x_5 x_4 x_3 x_7 x_2
x_6	x_4 x_8	x_6	x_{10} x_8
x_1		x_1	
x_9		x_5	
$\sum P_i$	26.68	$\sum P_i$	26.68

Tabla 5.5 : Cuadros de agrupamiento de incógnitas.

La primera incógnita a ser seleccionada como semilla debe ser aquella cuyo agrupamiento posea la mayor sumatoria de pesos. Sin embargo, se observa en la tabla 5.5 que los agrupamientos de las 4 incógnitas candidatas a semillas tienen igual sumatoria de pesos. Entonces, en este ejemplo se seleccionará a la incógnita x_6 como primera semilla, por ser la primera de las candidatas (mayor peso individual). Así, la incógnita x_6 es incluida en el conjunto de semillas ***S***.

Como las condiciones del problema especificaban que éste debe ser particionado en dos subsistemas, se necesitarán por lo tanto dos semillas. La segunda semilla deberá ser aquella candidata (aún no seleccionada) cuya sumatoria de pesos ($\sum P_i$) del conjunto ***I*** sea la segunda mayor. En este caso, el conjunto ***I*** de la incógnita x_1 cumple con esta condición, pero no puede ser la segunda semilla porque no cumple con la condición de radio de vecindad mínimo determinada por el parámetro *nvec*, que estipula que una semilla no debe encontrarse entre las *nvec* primeras incógnitas asociadas del agrupamiento de incógnitas de las semillas previamente seleccionadas. En el agrupamiento de la semilla x_6 previamente escogida, se encuentra a la incógnita x_1 como segunda asociada, lo que la imposibilita de ser seleccionada como segunda semilla. También se puede observar que la siguiente candidata, x_5 se encuentra como primera asociada, lo que la imposibilita de ser seleccionada como segunda semilla. La candidata restante se encuentra como tercera asociada en el agrupamiento de x_6, por lo que se la selecciona como segunda semilla incluyéndola en el conjunto ***S***. De este modo, ya se dispone de un conjunto de semillas a partir del cual generar la descomposición del sistema de ecuaciones, esto es

$$\boldsymbol{S} \_ \{x_6, x_9\}$$

5.4 Generación de la partición

Una vez determinadas las semillas para cada subsistema de acuerdo al algoritmo de selección de semillas, es necesario adicionar las demás incógnitas a los subsistemas más adecuados. Se podrá tener tantos conjuntos ***S*** como ternas *vlim*, *ngrup* y *nvec* se tengan. Para fines didácticos, se cambiará el conjunto ejemplo $\boldsymbol{S}\_\{x_6, x_9\}$ y se tomará el conjunto $\boldsymbol{S}\_\{x_3, x_1\}$ con vista a obtener un proceso de partición con la mayor cantidad posible de peculiaridades.

El algoritmo de generación de la partición posee alguna similitud con el algoritmo de selección de semillas. La diferencia principal consiste en que en la selección de semillas, la asociación de incógnitas se lleva a cabo en forma independiente, sin existir interacción entre los distintos agrupamientos, mientras que en la generación de la partición todos los agrupamientos alrededor de las distintas semillas se realizan en forma simultánea, interactuando entre sí.

La tabla 5.6 que se muestra a continuación es bastante similar a las utilizadas en la sección 5.3. La columna ***J*** ahora contendrá al subsistema en formación. Se llamará subsistema 1 a la formada en torno a la semilla x_3, y subsistema 2 a la correspondiente a la semilla x_1. Debe observarse que en las columnas ***CIA*** que contienen a las incógnitas adyacentes al subconjunto en formación no se encuentran las demás semillas, por más que estas se encuentren entre las adyacentes. Esto se debe a que en la citada columna solo se indicarán las incógnitas adyacentes y disponibles de ser asociadas, es decir, las incógnitas aun no asociadas por ningún subsistema.

Subsistema 1		**Subsistema 2**	
J_3	CIA_3	J_1	CIA_1
x_3	$x_6\, x_5\ x_9\ x_4\ x_{10}\ x_2$	x_1	$x_5\ \ x_9\ \ x_4\ \ x_{10}\ \ x_7\ \ x_2\ \ x_8$

Tabla 5.6 : Cuadro de generación de la partición: situación 1.

Se impone aquí el cálculo del vector **Q** que se utilizará para determinar cuáles subsistemas en formación anexarán incógnitas en un instante dado. De acuerdo a lo expuesto en la sección 5.4, las componentes del vector **Q** se calculan dividiendo las componentes del vector **C** (cantidad de incógnitas agrupadas en el subsistema en un momento dado) por las componentes del vector **w** (vector de *performances* relativas). El vector **Q** será entonces:

$$\mathbf{Q} = [q_1,\ q_2] = \left[\frac{c_1}{w_1},\ \frac{c_2}{w_2}\right] = \left[\frac{1}{4},\ \frac{1}{1}\right] = [0.25,\ 1]$$

Teniendo en cuenta las dimensiones que se desean para los subsistemas, lo ideal sería que todas las componentes del vector **Q** sean iguales entre sí. Se ha detectado así que el subsistema 1 se encuentra en desventaja con respecto al otro en lo que se refiere a la relación que debe existir entre sus dimensiones. En consecuencia, solamente el subsistema 1 tiene derecho a anexar incógnitas. Así, seleccionará la incógnita más pesada de entre sus adyacentes a fin de anexarla, en este caso la incógnita x_6, pasado ésta a formar parte automáticamente del subsistema 1. La nueva situación puede verse en la tabla 5.7, que muestra al subsistema 1 con 2 incógnitas (x_3 y x_6) y al subsistema 2 con 1 incógnita (x_1).

Subsistema 1		Subsistema 2	
J_3	CIA_3	J_1	CIA_1
x_3	~~x_6~~ x_5 x_9 x_4 x_{10} x_2	x_1	x_5 x_9 x_4 x_{10} x_7 x_2 x_8
x_6	x_7 x_8		

Tabla 5.7 : Cuadro de generación de la partición: situación 2.

Se observa también que en CIA_3 se han incluido las incógnitas x_7 y x_8, que son las nuevas incógnitas adyacentes al subsistema 1 a través de la incógnita x_6 .

El vector $\mathbf{Q}$ será ahora:

$$\mathbf{Q} = [q_1, q_2] = \left[\frac{c_1}{w_1}, \frac{c_2}{w_2}\right] = \left[\frac{2}{4}, \frac{1}{1}\right] = [0.5, 1]$$

Esto indica que aun el subsistema 1 se encuentra en desventaja frente al subsistema 2, tieniendo derecho a incorporar más incógnitas hasta que ambas se encuentren con las dimensiones relativas deseadas. Así, el subsistema 1 anexa 2 incógnitas más como se puede ver en la tabla 5.8.

El vector $\mathbf{Q}$ será ahora:

$$\mathbf{Q} = [q_1, q_2] = \left[\frac{c_1}{w_1}, \frac{c_2}{w_2}\right] = \left[\frac{4}{4}, \frac{1}{1}\right] = [1, 1]$$

Subsistema 1		Subsistema 2	
J_3	CIA_3	J_1	CIA_1
x_3	x_6 x_5 x_9 x_4 x_{10} x_2	x_1	x_5 x_9 x_4 x_{10} x_7 x_2 x_8
x_6	x_7 x_8		
x_5			
x_9			

Tabla 5.8: Cuadro de generación de la partición: situación 3

Dado que ambos subsistemas se encuentran con las dimensiones relativas deseadas, cada uno de ellos procede a seleccionar a la incógnita más pesada de su entorno. En este caso, se verifica una coincidencia de candidatas sobre la incógnita x_4. Se la debe asignar a aquel subconjunto con el cual posea el mayor acoplamiento. Se observa en la matriz **M** que $m_{3\,4} = m_{1\,4} = 1$, es decir, los acoplamiento de esta incógnita con ambos subsistemas tienen igual valor. En este punto se puede tomar dos caminos: realizar o no solapamiento parcial.

En el caso no hacer solapamiento parcial, la incógnita x_4 pasa a formar parte de J_3. En realidad, ésta puede ser asignada a cualquiera de los subconjuntos que compiten por ella, pero se optó como mejor opción asignar la incógnita en competencia al procesador que tiene el mayor valor de *performance* relativa (en este caso el subconjunto 1). Lo mismo ocurre con la siguiente incógnita candidata, x_{10}. La siguiente candidata es x_7 , que se encuentra en ambas ***CIA***s, y como $m_{4\,7} = 2 > m_{1\,7} = 1.2$, ella pasa a formar parte de J_3. El procedimiento continúa hasta que todas las incógnitas hayan sido incluidas por algún conjunto, completándose la descomposición. La partición resultante está dada por los conjuntos $\{x_3, x_6, x_5, x_9, x_4, x_{10}, x_7 , x_2\}$ y $\{x_1, x_8\}$ (tabla 5.9.a), que satisface la condición $\mathbf{w} = [4\ 1]^T$.

Subsistema 1		**Subsistema 2**	
J_3	CIA_3	J_1	CIA_1
x_3	x_6 x_5 x_9 x_4 x_{10} x_2	x_1	x_5 x_9 x_4 x_{10} x_7 x_2 x_8
x_6	x_7 x_8	x_8	
x_5			
x_9			
x_4			
x_{10}			
x_7			
x_2			

Tabla 5.9.a: Cuadro de generación de la partición.
Caso 1: Sin solapamiento parcial.

Se analiza ahora la opción con posibilidad de solapamiento. Puesto que existe una contienda por la incógnita x_4, debemos comparar el peso de la misma con el límite de solapamiento (asumimos *LimOver* = 6). Como $P_4 = 6.21 \geq$ *LimOver*, el método escoge utilizar solapamiento; entonces la incógnita x_4 es incluida en ambos subsistemas. Habiendo J_2 completado su cupo parcial, las incógnitas x_{10}, x_7 y x_2 pasan automáticamente a formar parte de J_2. A continuación, ambos conjuntos seleccionan a x_8 como siguiente incógnita a ser incluida. Nuevamente ocurre una contienda, ya que $m_{10\,8} \_ m_{1\,8} \_ 2$, pero como $P_8 = 4.61 <$ *LimOver*, la incógnita x_8 es anexada al subsistema 1. La partición resultante está dada por los conjuntos $\{x_3, x_6, x_5, x_9, x_4, x_{10}, x_7, x_2, x_8\}$ y $\{x_1, x_4\}$ (tabla 5.9.b.), con $\mathbf{w} = [4.5 \;\; 1]^T$. Note que x_4 pertenece a ambos subsistemas.

Subsistema 1		**Subsistema 2**	
J_3	CIA_3	J_1	CIA_1
x_3	x_6 x_5 x_9 x_4 x_{10} x_2	x_1	x_5 x_9 x_4 x_{10} x_7 x_2 x_8
x_6	x_7 x_8	x_4	
x_5			
x_9			
x_4			
x_{10}			
x_7			
x_2			
x_8			

Tabla 5.9.b: Cuadro de generación de la partición.
Caso 2: Con solapamiento parcial.

5.5 Evaluación de particiones y selección

El parámetro de selección propuesto en este trabajo es el *radio espectral* de la matriz de comparación. Esta matriz, de singular importancia en el estudio de los métodos bloque iterativos, se forma en función del método utilizado en la resolución del problema. Aún cuando el método utilizado en el marco de este trabajo para resolver sistemas de ecuaciones es una variante del método de Newton-Raphson, por simplicidad en la explicación de este ejemplo se supondrá aquí que el sistema se resolverá aplicando el método de Jacobi (sección 2.1).

Conforme a lo presentado en la sección 2.1 , el algoritmo iterativo de Jacobi para la resolución de sistemas lineales de ecuaciones puede ser expresado como:

$$\mathbf{x}(k+1) = \mathbf{S}^{-1}\mathbf{T}\,\mathbf{x}(k) = \begin{bmatrix} \mathbf{S}_{11} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{S}_{pp} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{T}_{11} & \cdots & \mathbf{T}_{1p} \\ \vdots & \ddots & \vdots \\ \mathbf{T}_{p1} & \cdots & \mathbf{T}_{pp} \end{bmatrix} \mathbf{x}(k) \qquad (5.1)$$

Los elementos de la matriz de comparación **H** se calculan tomando normas de los bloques de la matriz de iteración $\mathbf{S}^{-1}\mathbf{T}$. Para el ejemplo considerado, y teniendo en cuenta la partición obtenida sin realizar solapamiento, se tendrá entonces que

$$\mathbf{H} = \begin{bmatrix} \left\| \mathbf{S}_{11}^{1}\mathbf{T}_{11} \right\| & \left\| \mathbf{S}_{11}^{1}\mathbf{T}_{12} \right\| \\ \left\| \mathbf{S}_{22}^{1}\mathbf{T}_{21} \right\| & \left\| \mathbf{S}_{22}^{1}\mathbf{T}_{22} \right\| \end{bmatrix}, \qquad \mathbf{H} \in \mathbb{R}^{2\times 2} \tag{5.2}$$

y considerando para mayor simplicidad que $\mathbf{T}_{11}=\mathbf{T}_{22}=\mathbf{0}$

$$\mathbf{H} = \begin{bmatrix} 0 & \left\| \mathbf{S}_{11}^{1}\mathbf{T}_{12} \right\| \\ \left\| \mathbf{S}_{22}^{1}\mathbf{T}_{21} \right\| & 0 \end{bmatrix} = \begin{bmatrix} 0 & h_{12} \\ h_{21} & 0 \end{bmatrix} \tag{5.3}$$

entonces

$$\rho(\mathbf{H}) = \sqrt{h_{12} \cdot h_{21}} \tag{5.4}$$

Así, el cálculo del radio espectral de la matriz **H** se redujo en este caso a calcular la raíz cuadrada del producto de los elementos no diagonales de la matriz.

El cálculo arriba ejemplificado se debería repetir con cada una de las particiones que se desea comparar, de forma a obtener el radio espectral $\rho(\mathbf{H})$ para cada una de las particiones obtenidas. De este modo, se considerará como mejor partición aquella que posea el menor valor del radio espectral de la matriz de comparación, de entre todas las generadas por el método partiendo de distintos grupos de semillas y considerando o no la posibilidad de solapamiento parcial.

Así, para las dos particiones del ejemplo se tiene:

partición 1: $\{x_6, x_5, x_1, x_4, x_{10}, x_7, x_2, x_8\}, \{x_9, x_3\}$ $\rho(\mathbf{H}) = 0.0709$

partición 2: $\{x_3, x_6, x_5, x_9, x_4, x_{10}, x_7, x_2\}, \{x_1, x_8\}$ $\rho(\mathbf{H}) = 0.101465$

Entonces, según lo expresado en la sección 4.5, la partición número 1 es la seleccionada como mejor partición.

CAPITULO 6: DISEÑO DEL PROGRAMA DE PARTICIONES

6.1 Introducción

Una vez definido el método (ver capítulo 4) a ser utilizado para descomponer un sistema de ecuaciones en subsistemas menores, se diseñó el programa generador de descomposiciones. En la figura 6.1 se puede observar el diagrama que resume el comportamiento del programa.

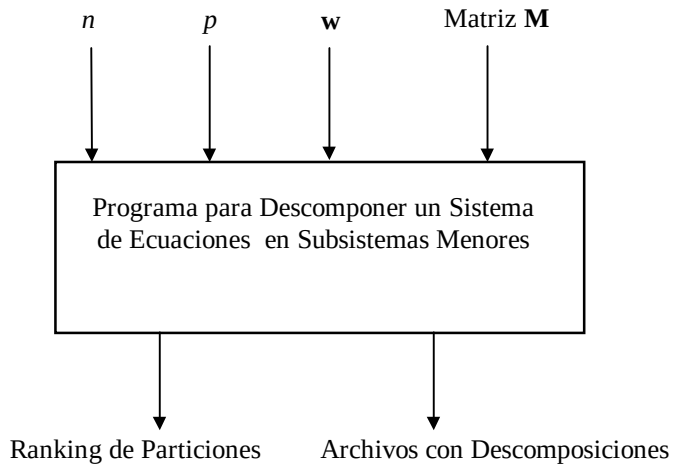


Figura 6.1 : Esquema del programa implementado.

En la concepción del programa se tuvo en cuenta, además, la posibilidad de interacción del usuario con el programa, pudiendo el mismo elegir las ternas o las semillas en la generación de la descomposición. Para esto, se buscó un entorno que ofreciera mejores recursos en la interfaz hombre-máquina, seleccionándose el ambiente Windows y utilizándose el software Borland C++ Versión 4.0 para la codificación del programa (ver Apéndice B).

Así mismo, se elaboró una versión simplificada del programa en lenguaje ANSI C, de manera a permitir la migración del mismo a plataformas computacionales mas poderosas.

La metodología seguida consistió en el desarrollo de diagramas de flujo de datos (DFD), el diseño de pantallas y la codificación del programa. En las secciones siguientes del capítulo se detallan cada uno de estos pasos.

6.2 DFD

El diagrama de flujo de datos se realizó siguiendo la nomenclatura utilizada en [31]. Se optó por esta forma de análisis estructurado debido a que ella no necesita de explicaciones, ya que es una representación simple y bastante intuitiva. Se presentan así el Diagrama de Contexto y dos niveles donde se detallan cada uno de los procesos o funciones conforme a una numeración secuencialmente asignada, así como las entradas, salidas, flujo interno de datos y los archivos utilizados, ordenados en forma alfabética.

6.2.1 Nivel 0: Diagrama de contexto

ENTRADAS:

Dato del Archivo del Sistema de Ecuaciones:
Es el nombre del archivo donde está la matriz **M** del sistema de ecuaciones que se desea descomponer.

Datos del Sistema Distribuido:
Son los datos referentes al sistema distribuido que se utilizará para resolver el sistema de ecuaciones; específicamente, el número de procesadores (*p*) del

sistema distribuido y la *performance* relativa entre los mismos (**w**)

Opciones:

Son los datos que el usuario selecciona para realizar la descomposición del sistema de ecuaciones. Éstos son: OpciónSelTer, OpciónSelSem, OpciónOver y OpciónParam.

Las opciones en cuanto a generación de descomposiciones que se pueden dar son:

- selección de ternas, selección de semillas y descomposición realizadas de forma automática, o
- selección de ternas por el usuario, o
- selección de semillas por el usuario.

También el usuario debe poder elegir el parámetro de selección de descomposiciones a ser utilizado ("Par_A" o $_{b}$ (**H**)) y poder optar por realizar o no solapamiento parcial.

ALMACENAMIENTOS

Archivo del Sistema de Ecuaciones:

Es el archivo donde se encuentra la matriz **M** del sistema de ecuaciones, que como se indicó en la sección 4.1, es la matriz que indica los acoplamientos existentes entre las incógnitas del sistema de ecuaciones. Ésta puede ser, por ejemplo, la matriz de coeficientes del sistema o una versión reordenada de la misma.

Archivos de Descomposiciones:

Aunque en el DFD hay un solo símbolo, realmente se tienen tantos archivos como descomposiciones sean generadas. Cada archivo contiene los subsistemas que forman parte de una descomposición del sistema de ecuaciones.

PROCESOS O FUNCIONES:

Descomponer un Sistema de Ecuaciones en Subsistemas Menores:

Es el proceso que descompone un sistema de ecuaciones (con posibilidad de interacción con el usuario) de forma tal que su resolución paralela en un sistema distribuido heterogéneo sea eficiente.

FLUJO DE DATOS:

ArchDes:

Es el archivo destino donde se encuentran los datos de una descomposición generada.

Dimensión del Sistema de Ecuaciones:

Es la dimensión del sistema de ecuaciones a ser descompuesto.

Matriz M:

Es la matriz cuyos elementos m_{ij} y m_{ji} representan al acoplamiento existente entre las incógnitas x_i y x_j (ver sección 4.1).

Nombre del Archivo:

Es el nombre del archivo que contiene a la matriz **M** del sistema de ecuaciones.

Número de Procesadores:

Es el número p de procesadores que componen el sistema distribuido donde se implementará la resolución paralela del sistema de ecuaciones.

OpciónOver:

Indica si se desea o no que el programa realice el solapamiento parcial de las ecuaciones críticas (ver sección 2.5).

OpciónParam:

Indica cual de los dos parámetros de selección de particiones ("Par_A" o b (**H**) de la

sección 4.5) se tendrá en cuenta como criterio de selección de descomposiciones.

OpciónSelSem (Opción de Selección de Semillas)**:**
Bandera que indica la forma de selección de semillas. Indica si el usuario desea que el programa seleccione las semillas o que éstas serán seleccionadas por el usuario en forma manual.

OpciónSelTer (Opción de Selección de Ternas)**:**
Bandera que indica la forma de selección de ternas. Indica si el usuario desea la selección automática de ternas o si éstas serán seleccionadas manualmente por el usuario.

Partición:
Son los nombres de los archivos que contienen las descomposiciones generadas por el programa.

PerformRelativa:
Es la *performance* relativa existente entre cada uno de los procesadores del sistema distribuido (ver sección 4.4).

ValorPar:
Son los valores del parámetro de selección de descomposiciones obtenidos a partir de cada una de las descomposiciones generadas por el programa.

SALIDA:

Ranking de particiones:

Se presenta un ranking de las descomposiciones generadas por el método junto con los valores del parámetro de selección de descomposiciones, pudiéndose identificar aquella que presente el menor valor del parámetro de selección, es decir, la mejor descomposición del sistema de ecuaciones sugerida por el programa al usuario.

6.2.2 Nivel 1: Programa General

ENTRADAS (ver definiciones en la sección 6.2.1):

Dato del Archivo del Sistema de Ecuaciones

Datos del Sistema Distribuido

Opciones

ALMACENAMIENTOS (ver definiciones en la sección 6.2.1):

Archivo del Sistema de Ecuaciones

Archivos de Descomposiciones

PROCESOS O FUNCIONES:

1 Interface:

Es el proceso donde tiene lugar la interface con el usuario. Aquí se determinan los parámetros del programa así como los pasos a seguir de acuerdo a las opciones que el usuario ha especificado.

2 Clasificación de Incógnitas:

Es el proceso de clasificación de las incógnitas, el cual se realiza siguiendo el algoritmo de clasificación de incógnitas presentado en la sección 4.2 (figura 4.2). Esta clasificación se basa en un ranking de pesos, donde el peso es una medida ponderada del grado de acoplamiento de las incógnitas.

3 Selección de Ternas:

Es el proceso de selección de las ternas *vlim*, *ngrup*, *nvec* a ser utilizadas para generar las posibles particiones (ver sección 4.3). Este proceso es muy importante porque de acuerdo a cuántas y cuáles sean las ternas generadas se tendrá la cantidad y la calidad de las particiones.

4 Selección de Semillas:

Es el proceso de elegir las semillas necesarias a partir de las cuales generar la partición. Se realiza según el algoritmo de selección de semillas presentado en la sección 4.2 (figura 4.3). Este proceso se realiza tantas veces como número de ternas haya.

5 Formación de Particiones:

Es el proceso en el cual, a partir de las semillas, se generan los subconjuntos de incógnitas que formarán parte de cada partición. Se realiza según el algoritmo de partición presentado en la sección 4.2 (figura 4.4). Este proceso se realiza tantas veces como conjuntos diferentes de semillas haya.

6 Evaluación de Particiones y Selección

Es el proceso donde el programa realiza un ranking de las particiones generadas. Utilizando el criterio de selección deseado determina cual de las particiones generadas ofrece las mejores características en términos del desempeño computacional de los métodos de resolución bloque iterativos. Este proceso se realiza según el algoritmo de selección de particiones presentado en la sección 4.2

(figura 4.5).

FLUJO DE DATOS

EXTERNO (ver definiciones en la sección 6.2.1):

ArchDes

Dimensión del Sistema de Ecuaciones

Matriz M

Nombre del Archivo

Número de Procesadores

OpciónOver

OpciónParam

OpciónSelSem

OpciónSelTer

Partición

PerformRelativa

ValorPar

INTERNO:

LimOver:

Se define *LimOver* como el mínimo valor requerido del peso de una incógnita para realizar solapamiento parcial (ver sección 2.5). Si el peso P_i de un una incógnita en disputa es mayor o igual que *LimOver,* entonces se realiza el solapamiento parcial de esa incógnita.

Otra:

Indica que se desea otra terna, ya que la actual generó un grupo de semillas ya generado anteriormente. Esto es necesario considerar debido al hecho ternas *vlim*,

ngrup, *nvec* distintas pueden generar el mismo grupo de semillas y consecuentemente generarán la misma partición.

Over:

Indica que el usuario ha optado por realizar el solapamiento parcial en los casos en que fuera necesario (ver sección 2.5), de acuerdo al valor de LimOver.

ParticiónGenerada:

Indica cuales incógnitas están en cada una de las particiones generadas.

Pesos:

Son los pesos de las incógnitas del sistema de ecuaciones calculados según la ecuación (4.2) de la sección 4.2.

Semillas:

Son las semillas (conjunto ***S*** de la sección 4.3) que generaran la descomposición del sistema de ecuaciones.

SemUsuario:

Indica las incógnitas elegidas por el usuario para ser semillas de la descomposición.

Siguiente:

Indica que se debe tomar otra terna de entre las generadas automáticamente para generar otra descomposición.

Terna:

Es la terna *vlim*, *ngrup*, *nvec* (ver sección 4.3) generada en forma automática.

TernaUsuario:

Es la terna *vlim*, *ngrup*, *nvec* seleccionada por el usuario.

SALIDA (ver definiciones en la sección 6.2.1):

Ranking de particiones

6.2.3 Nivel 2

Se presentan a continuación los DFD's de los seis procesos del DFD de nivel 1 de la figura 6.3.

6.2.3.1 Interface

PROCESOS O FUNCIONES:

1.1 Decidir pasos:

Es el proceso donde según las opciones del usuario se deciden los pasos que se van seguir.

1.2 Pedir Terna:

Es el proceso donde el programa pide al usuario la terna de valores *vlim*, *ngrup*, *nvec* (ver sección 4.3) que será utilizada para la selección de semillas, en el caso que el usuario haya optado por la selección manual de ternas. .

1.3 Pedir Semillas:

Es el proceso que solicita al usuario las semillas a ser utilizadas para generar una partición, en caso que el usuario haya optado por la selección manual de semillas.

1.4 Pedir LimOver:

Es el proceso que solicita al usuario un límite inferior de peso requerido para realizar el solapamiento parcial (ver sección 2.5). El programa sugerirá al usuario un valor LimOver predeterminado, pero el usuario es libre de cambiarlo si así lo prefiere.

FLUJO DE DATOS:

EXTERNO (ver definiciones en la sección 6.2.2):

LimOver

OpciónOver

OpciónSelSem

OpciónSelTer

Over

SemUsuario

TernaUsuario

INTERNO:

OpManual:

Indica que el usuario seleccionará la terna *vlim*, *ngrup*, *nvec* en forma manual.

OpSem:

Indica que el usuario determinará en forma manual las *p* semillas requeridas.

OpOver:

Indica que el usuario desea la realización de solapamiento parcial de incógnitas en el proceso de formación de las descomposiciones.

6.2.3.2 Selección de semillas

PROCESOS O FUNCIONES:

4.1 Selección de Candidatas a Incógnitas Semillas:

Es el proceso de elegir cuáles incógnitas tienen un "peso" mayor o igual al valor de *vlim* de manera a ser incluidas como candidatas a semillas, pasando a formar parte del conjunto ***K*** de la sección 4.3.

4.2 Formación de Subconjuntos de las Incógnitas Candidatas a Semillas:

En este proceso se realiza la anexión de *ngrup* incógnitas alrededor de cada candidata a semilla (los conjuntos ***I*** de la sección 4.3) para luego calcular la sumatoria de los pesos de cada uno de los subconjuntos.

4.3 Evaluación de las Candidatas:

Una vez que se tienen las sumatorias de pasos (sección 4.3) se eligen las *p* sumatorias mayores para seleccionarlas como semillas, siempre teniendo en cuenta el valor del parámetro *nvec*.

4.4 Ver Semillas Repetidas:

Es el proceso de ver si una terna dada genera semillas ya generadas anteriormente por una terna diferente previamente seleccionada.

FLUJO DE DATOS:

EXTERNO (ver definiciones en la sección 6.2.2):

Matriz M

Número de procesadores

Otra

Pesos

Semillas

Terna (*vlim*, *ngrup*, *nvec*)

TernaUsuario (*vlim*, *ngrup*, *nvec*)

INTERNO:

Candidatas:

Son las incógnitas candidatas a ser seleccionadas como semillas (conjunto ***K*** de la sección 4.3).

NoRepetidas:

Es una bandera que indica si la terna actual no genera semillas candidatas ya seleccionadas por otra terna.

Semillas:

Son las *p* semillas necesarias para la formación de una partición (conjunto ***S*** de la sección 4.3).

Sumatoria de Pesos:

Es la sumatoria de los pesos de los subconjuntos formados alrededor de las incógnitas a semillas.

6.2.3.3 Formación de Particiones

PROCESOS O FUNCIONES:

5.1 Determinación de los Vectores de Estado:

Es el proceso que calcula los dos vectores de estado **Q** y **C**. Como se explicó en la sección 4.4., estos vectores son necesarios para controlar que el balanceamiento se realice de acuerdo a la *performance* relativa **w** existente entre

los procesadores del sistema distribuido.

5.2 Formación de Subconjuntos:

Es el proceso de formación de los subconjuntos ***J*** (ver sección 4.4.) de las incógnitas, donde cada uno de los subconjuntos estará compuesto de las incógnitas que corresponderán a un subsistema a ser resuelto por un procesador del sistema distribuido. Al principio, cada subconjunto contendrá a una semilla para luego se ir anexando las demás incógnitas hasta formar la partición.

5.3 Determinación de los Subconjuntos que Pelean:

Es el proceso por el cual se determinan cuales son los subconjuntos que competirán por anexar una incógnita. Este proceso es el que controla que el balanceamiento se lleve a cabo según la *performance* relativa **w** (ver sección 4.4) entre los procesadores del sistema distribuido.

5.4 Determinación de las Siguientes Incógnitas a Agrupar:

Es el proceso por el cual se decide cuáles son las siguientes incógnitas candidatas a ser anexadas a cada uno de los subconjuntos existentes. Para cada uno de ellos, se identifica entre los conjuntos de incógnitas adyacentes (***CIA***'s) cual es la incógnita de mayor peso que no haya sido seleccionada anteriormente, seleccionándola como candidata a ser anexada por ese subconjunto.

5.5 Verificación de Coincidencias:

Es el proceso que verifica si dos o más subconjuntos quieren anexar a una misma incógnita.

5.6 Pelea por la Incógnita Seleccionada:

Dado el caso de que dos o más subconjuntos en formación quieran anexar una misma incógnita, este proceso determina a cual de ellos es más propicio anexar la incógnita en cuestión. Si existen coincidencias de acoplamientos y si el usuario ha optado por la opción de solapamiento parcial, se pasa al siguiente proceso para realizar el mismo.

5.7 Solapamiento:

Es el proceso por el cual una incógnita, dependiendo de su peso y del valor de *LimOver*, es anexada simultáneamente a dos o más subconjuntos en formación.

FLUJO DE DATOS:

EXTERNO (ver definiciones en la sección 6.2.2):

LimOver

Matriz M

Over

ParticiónGenerada

PerformRelativa

Pesos

Semillas

INTERNO:

C, Q:

Son los vectores de estado necesarios para controlar el balanceamiento de carga. **C** indica la cantidad de incógnitas agrupadas en un momento dado y **Q** indica el cupo parcial (ver sección 4.4).

CIA's:

Son los Conjuntos de Incógnitas Adyacentes a los subconjuntos en formación.

IncógCand, VSubconj:

IncógCand (Incógnitas Candidatas) indica cuáles son las incógnitas candidatas a ser incluidas en los distintos subconjuntos que tienen derecho a anexar una incógnita. VSubconj es el vector de dichos subconjuntos.

Incógnita, SubconjCoincidentes:

Indica la incógnita candidata ser anexada y cuáles subconjuntos coinciden en el deseo de anexar la incógnita referida.

Incógnita, SubconjO:

Indica la incógnita a ser anexada y cuales subconjuntos anexarán a la incógnita referida en caso de solapamiento parcial (SubconjO = Subconjuntos para el *Overlapping* o Solapamiento parcial). Obsérvese si el peso de la incógnita es menor o igual que LimOver, no debe realizarse solapamiento y entonces SubconjO solo será un subconjunto.

Incógnita, Subconjunto:

Indica la incógnita x_i a ser anexada y a cuál subconjunto $\boldsymbol{J}$ se anexará.

QuienesPelean:

Indica cuales son los subconjuntos que tiene derecho a anexar una incógnita.

Subconjunto/s:

Son los subconjuntos que anexaron una incógnita. Cada vez que una incógnita es anexada se deberá indicar cual o cuales subconjuntos (en caso de solapamiento) han anexado la incógnita para que se actualicen los vectores de estado.

6.2.3.4 Evaluación de Particiones y Selección

PROCESOS O FUNCIONES:

6.1 Reordenación de la Matriz M:

Es el proceso por el cual la matriz **M** del sistema de ecuaciones es reordenada según lo indiquen los subconjuntos que forman una partición dada para luego calcular el parámetro de selección de descomposiciones.

6.2 Verificar Parámetro de Selección:

Es el proceso por el cual se verifica cuál de los dos parámetros posibles de selección de particiones fue seleccionado por el usuario para ser utilizado como criterio de selección de descomposiciones.

6.3 Determinación de Ramas de Corte:

Es el proceso que determina cuáles de los acoplamientos existentes entre los subconjuntos que forman una partición se cortan.

6.4 Formación de la Matriz de Comparación:

Es el proceso por el cual se forma la matriz de comparación.

6.5 Suma de las Ramas de Corte:

Es el proceso que suma los valores asociados todas las ramas de corte de una partición dada.

6.6 Cálculo del Radio Espectral:

Es el proceso que calcula el radio espectral de la matriz de comparación.

6.7 Guardar Valores:

Es el proceso que almacena los valores de la partición generada en un archivo, así como el nombre del archivo y el valor del parámetro de selección de descomposiciones

6.8 Decidir Siguiente Terna:

Es el proceso en el cual se decide si se debe generar otra posible partición (en el caso de la selección automática de ternas).

FLUJO DE DATOS:

EXTERNO (ver definiciones en la sección 6.2.2):

ArchDes

Matriz M

OpciónSelSem

OpciónSelTer

OpciónParam

Partición

Partición Generada

Siguiente

ValorPar

INTERNO:

Matriz de Comparación:

Es la matriz de comparación construida a partir de la matriz **M** reordenada.

Matriz M Ordenada:

Es la matriz **M** reordenada de acuerdo a la partición generada. Es necesaria para

luego formar la matriz de comparación o determinar cuáles son las ramas o acoplamientos que se cortan.

MIC:

Bandera que indica que fue seleccionado el radio espectral de la matriz de comparación como criterio selectivo de descomposiciones.

RC:

Bandera que indica que fue seleccionado Par_A como criterio selectivo de descomposiciones.

Ramas de Corte:

Son los acoplamientos que se cortan entre los subconjuntos que forman una partición.

6.3 Diseño de pantallas

El segundo paso en el desarrollo del programa fue el diseño de las pantallas. Se buscó con el mismo que el programa sea lo más claro posible para el usuario. Se incluyeron en el mismo cuadros de ayudas explicativos de los tópicos de las diversas pantallas.

En la figura 6.8 se presenta el diagrama que describe el flujo que deben seguir las ventanas a medida que el usuario va introduciendo las opciones del programa.

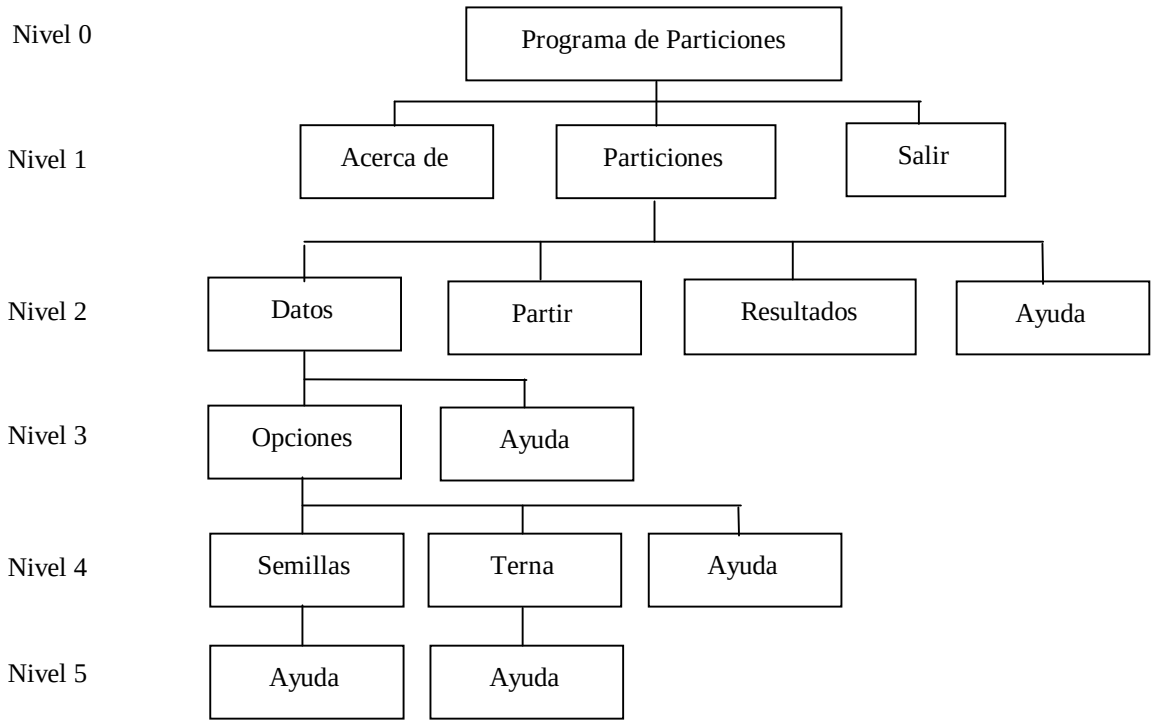


Figura 6.8 : Gráfico del flujo de pantallas.

A continuación se presenta cada pantalla y una breve explicación de cada una.

En la figura 6.9 se puede observar la pantalla **Programa de particiones** (nivel 0 de la figura 6.8) Esta pantalla presenta al usuario el menú del programa de particiones, donde se tienen las opciones: **Acerca de**, **Empezar** y **Salir**. La primera opción deberá desplegar una ventana que indique la información general del programa. La opción **Empezar** deberá desplegar una ventana en la que el usuario introduzca los datos del sistema que desea descomponer. Por último, seleccionando la tercera opción se abandonará el programa.

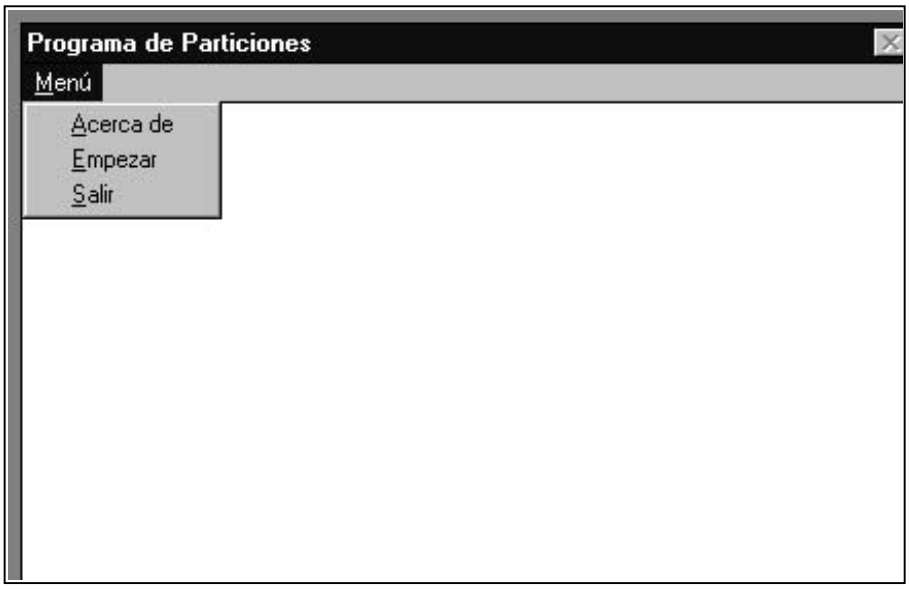


Figura 6.9 : Pantalla **Programa de Particiones** (nivel 0 de la figura 6.8).

En la figura 6.10 se observa la pantalla **Acerca de** (nivel 1 de la figura 6.8)**.** Como se indicó anteriormente, esta pantalla se presenta para que el usuario tenga una visión del objetivo del programa, así como de los datos generales del mismo.

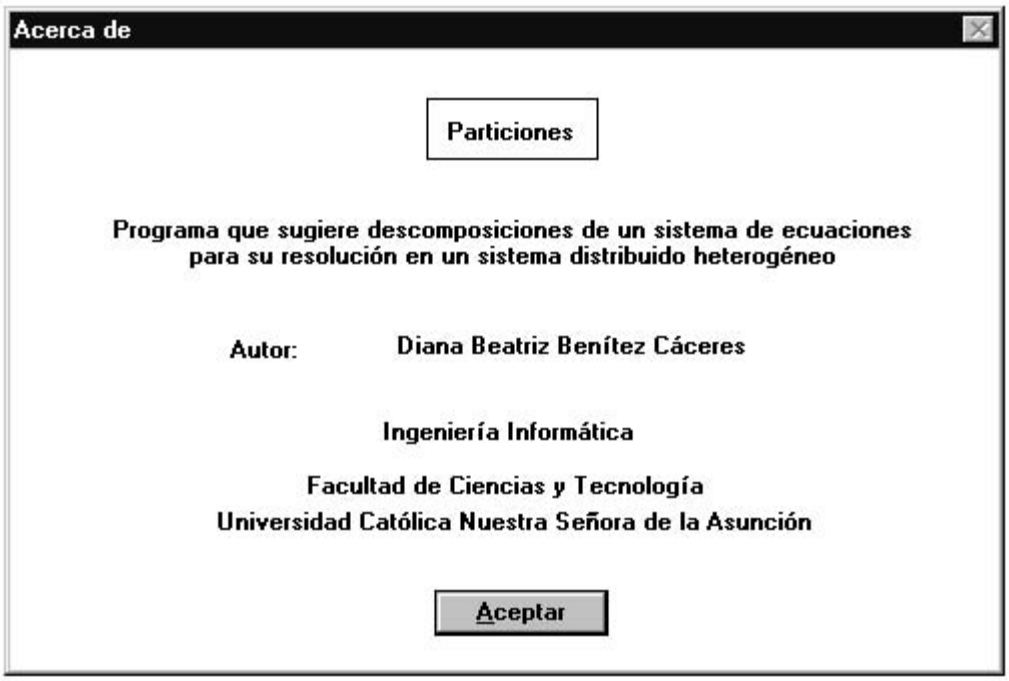


Figura 6.10 : Pantalla **Acerca de** (nivel 1 de la figura 6.8).

La pantalla **Particiones** de la figura 6.11 (nivel 1 de la figura 6.8) es la que se despliega cuando el usuario selecciona la opción **Empezar** del menú. Esta pantalla cuenta con cinco botones que son:

1- Datos : conduce a la pantalla **Datos** la figura 6.12 (nivel 2 de la figura 6.8), En ella se pedirán los datos del problema a ser descompuesto, como el nombre del archivo donde se encuentra la matriz del sistema, y la dimensión de la misma. También se solicitan los datos del sistema distribuido, es decir el número de procesadores que lo componen y la *performance* relativa de los mismos.

   En la pantalla **Datos** (nivel 2 de la figura 6.8) se puede observar el botón de Opciones que conduce a la pantalla **Opciones** de la figura 6.13 (nivel 3 de la figura 6.8) que es la que se utiliza para que el usuario determine sus opciones para la descomposición del sistema de ecuaciones.

2- Partir : Es la opción para realizar la partición del sistema según las indicaciones del usuario.

3- Resultados : Este botón conduce a la pantalla **Resultados** de la figura 6.16.a o a la pantalla **Resultado** de la figura 6.16.b, según sea el caso automático o manual (nivel 2 de la figura 6.8).

4- ? (Ayuda): Despliega una pantalla que dará una breve explicación de cada una de las opciones.

5- Salir :Cierra la ventana y vuelve al menú.

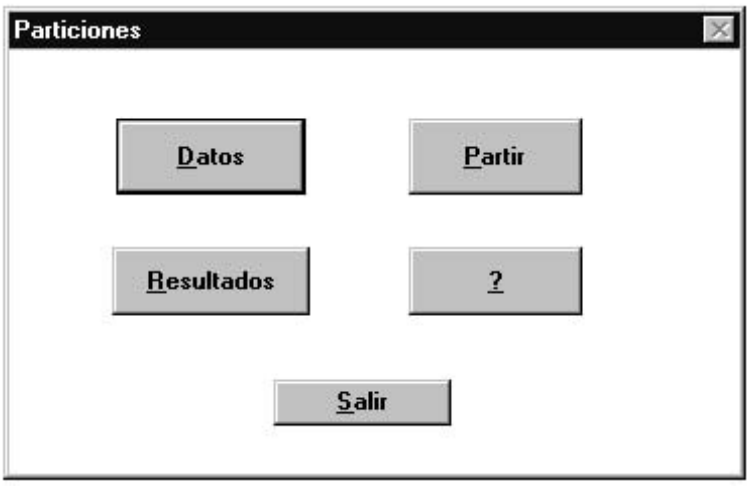


Figura 6.11 : Pantalla **Particiones** (nivel 1 de la figura 6.8).

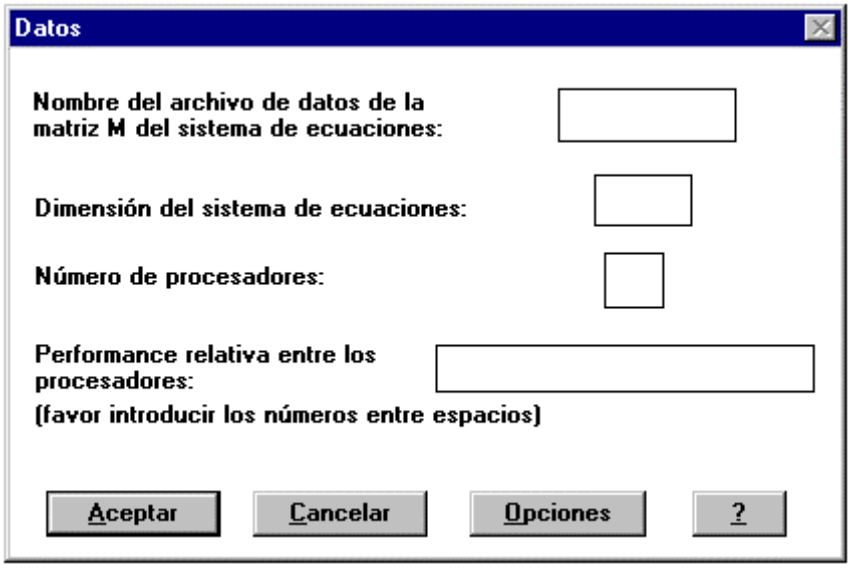


Figura 6.12 : Pantalla **Datos** (nivel 2 de la figura 6.8).

La pantalla **Opciones** de la figura 6.13 (nivel 3 de la figura 6.8) es la que el usuario

utiliza para establecer el tipo de descomposición a ser implementado. Como puede observarse en la figura 6.13, la selección de semillas, así como la selección de ternas, puede hacerse en forma manual (el usuario determina las semillas o las ternas) o en forma automática. El usuario también puede elegir el parámetro de selección de descomposiciones así como la opción de solapamiento parcial.

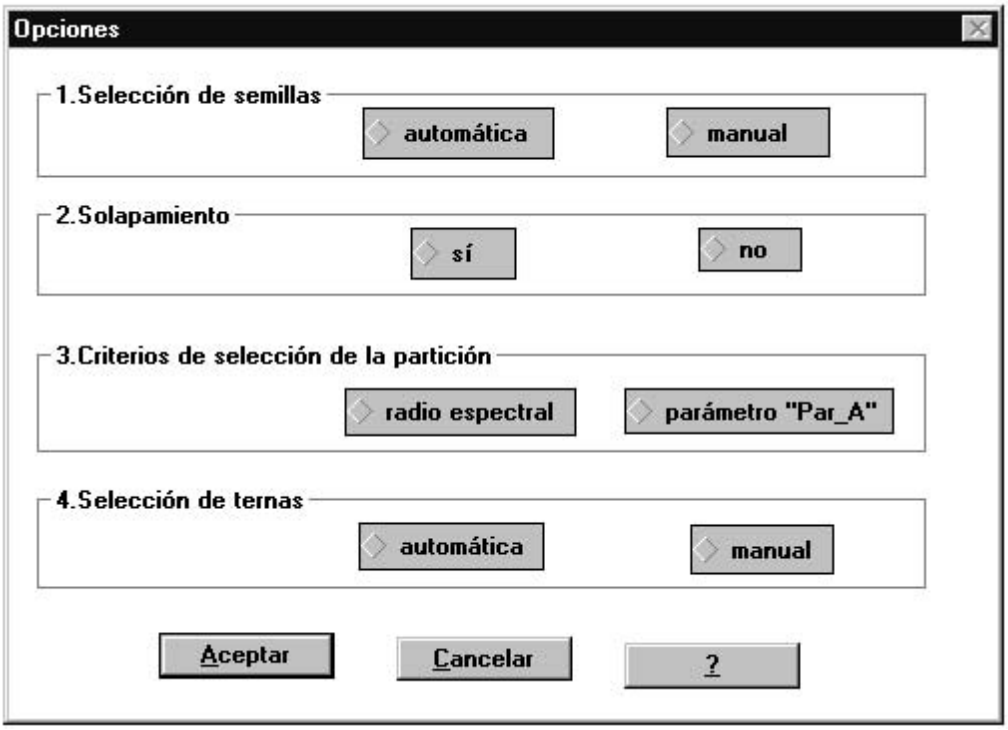


Figura 6.13 : Pantalla **Opciones** (nivel 3 de la figura 6.8).

En caso que el usuario opte por la selección manual de semillas, se habilita la pantalla **Semillas** de la figura 6.14 (nivel 4 de la figura 6.8.) y en ella el usuario puede indicar las incógnitas que serán utilizadas como semillas en la descomposición del sistema de ecuaciones.

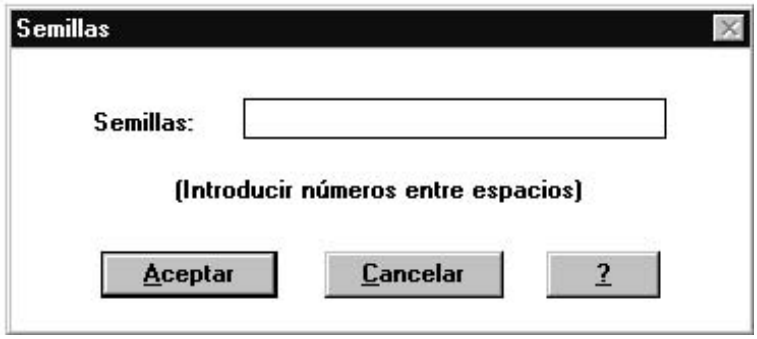


Figura 6.14.: Pantalla **Semillas** (nivel 4 de la figura 6.8.)

En la selección manual de ternas se tiene la pantalla **Terna del Usuario** de la figura 6.15 (nivel 4 de la figura 6.8).

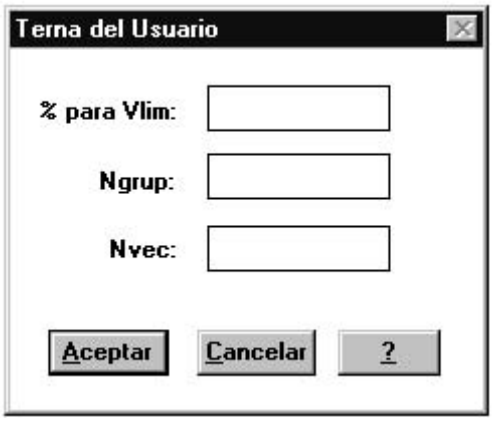


Figura 6.15: Pantalla **Terna del usuario** (nivel 4 de la figura 6.8).

La figura 6.16.a muestra la pantalla **Resultados** (nivel 2 de la figura 6.8) donde se presentan los resultados en el caso de que la descomposición se haya realizado en forma automática (es decir selección automática de ternas y de semillas). La figura 6.16.b muestra los resultados con la selección manual de ternas o de semillas.

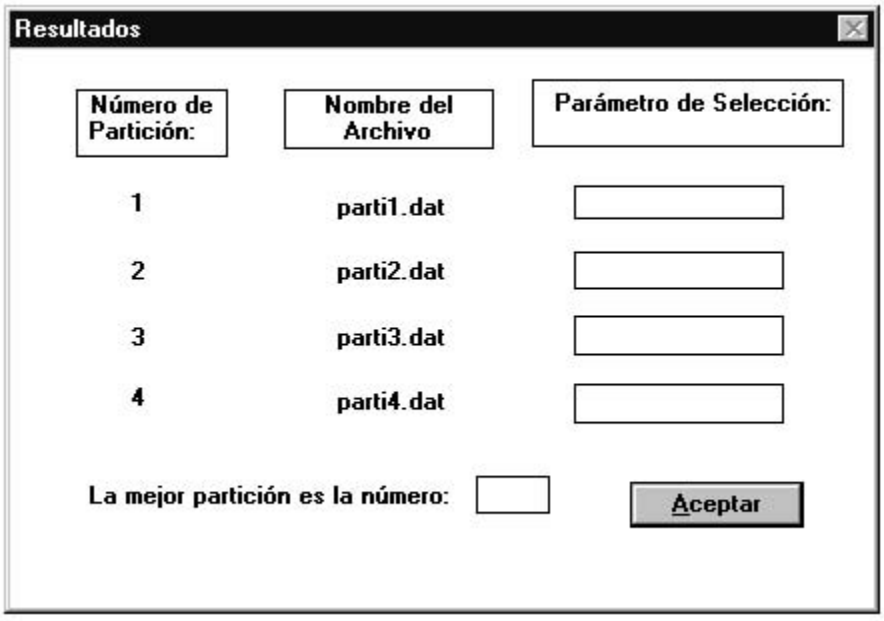


Figura 6.16.a : Pantalla **Resultados** con selección automática
(nivel 2 de la figura 6.8).

Figura 6.16.b : Pantalla **Resultado** con selección manual
(nivel 2 de la figura 6.8).

Por último, en cada una de las pantallas se dispone del botón de ayuda ? , que conduce a una pantalla donde se ofrece una breve explicación de la misma. Por ejemplo, el botón de ayuda de la pantalla **Particiones** (nivel 1 de la figura 6.8) conduce a la pantalla **Ayuda** (nivel 2 de la figura 6.8) que se puede observar en la figura 6.17.

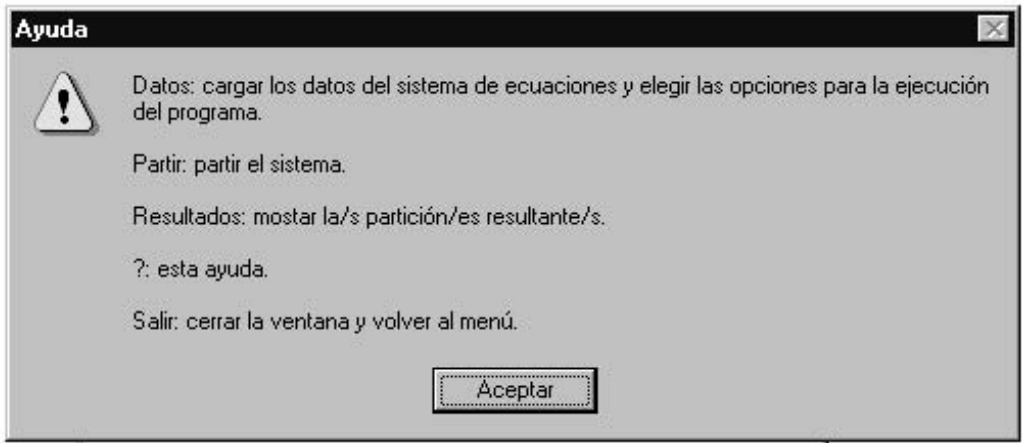


Figura 6.17 : **Ayuda** de la pantalla **Particiones**
(nivel 2 de la figura 6.8)

6.4 Pseudocódigo

El siguiente paso en el desarrollo del programa fue la elaboración del pseudocódigo, que se presenta en la figura 6.18. En el mismo se observan referencias a la sección 4, donde ya se han presentado los pseudocódigos de las diferentes etapas del proceso de descomposición.

Entradas: *n*, *p*, **w** y Matriz **M**.
n, es la dimensión del sistema de ecuaciones.
p, es el número de procesadores del sistema distribuido.
w, es el vector de *performance* relativa entre los procesadores del sistema distribuido
Matriz **M** del sistema de ecuaciones (ver sección 4.1.)

Introducir los datos iniciales y las opciones del usuario
Leer el archivo que contiene la matriz **M**
Clasificar las incógnitas según el algoritmo de clasificación de incógnitas de la figura 4.1.
SI la selección de ternas es automática **ENTONCES**
  Seleccionar las ternas en forma automática
  PARA cada terna seleccionada
    Seleccionar las semillas según el algoritmo de selección de semillas de la figura 4.2.
    Generar la partición según el algoritmo de formación de particiones de la figura 4.3.
    Almacenar la partición en un archivo
    Calcular el parámetro de selección de descomposiciones
  Hacer un ranking de las particiones generadas
SI la selección de terna es manual **ENTONCES**
  Seleccionar las semillas según el algoritmo de selección de semillas de la figura 4.2.
  Generar la partición según el algoritmo de formación de particiones de la figura 4.3.
  Almacenar la partición en un archivo
  Calcular el parámetro de selección de descomposiciones
SI la selección de semillas es manual **ENTONCES**
  Generar la partición según el algoritmo de formación de particiones de la figura 4.3.
  Almacenar la partición en un archivo
  Calcular el parámetro de selección de descomposiciones
Mostrar los resultados

Salidas: Ranking de particiones y Archivos de descomposiciones.

Figura 6.18 : Pseudocódigo del programa de particiones.

CAPITULO 7: ESTUDIOS EXPERIMENTALES

En este capítulo serán presentados los estudios experimentales realizados en base a problemas tipo de la IEEE (*The Institute of Electrical and Electronical Engineers*). En dichos estudios, se resolvió el problema del Flujo de Potencia Eléctrica (ver Apéndice B) para los sistemas eléctricos seleccionados como paradigmas.

Se analiza aquí el comportamiento de las particiones generadas por el método propuesto en este trabajo, a la vez de realizar un estudio comparativo con respecto a las particiones generadas por otros métodos y con otros criterios.

7.1 Ambiente computacional

El sistema distribuido utilizado consistió en una red Ethernet a 10 Mbps constituida por las siguientes estaciones de trabajo:

- Una workstation DEC 3000 modelo 300 con procesador ALPHA de 150 MHz, con 32 MB de memoria RAM y operando bajo el sistema operativo OSF/1 Versión 2.0.
- Una workstation SUN SPARC Station 5 con procesador SUN de 66 MHz y memoria RAM de 32 MB, operando bajo el sistema operativo Solaris 5.3.
- 10 Computadoras Personales PREMIO con procesadores Pentium de 75 MHz, 8 MB de memoria RAM y operando bajo el sistema operativo LINUX. Las máquinas fueron nombradas como: Linux1, Linux2,...,Linux10.

Los programas de resolución del Flujo de Potencia Eléctrica fueron implementados en lenguaje ANSI C, con la biblioteca PVM (Parallel Virtual Machine) Versión 3.10 en su extensión para lenguaje C.

Para el levantamiento de los datos experimentales se utilizaron dos sistemas

eléctricos tipos: el sistema IEEE de 14 barras (IEEE-14) y el sistema IEEE de 118 barras (IEEE-118). El primero de ellos fue seleccionado con vistas a poder realizar un estudio exhaustivo de manera a evaluar todas las posibles formas en que el sistema podría ser descompuesto. Dicho estudio sería imposible para sistemas de mayor dimensión debido a que el número de posibles particiones del sistema crece en forma factorial con la dimensión del problema.

7.2 Resolución del Sistema IEEE-14

La figura 7.1 presenta al Sistema Eléctrico IEEE-14. En realidad, el sistema de ecuaciones que representa el problema no incluye a la incógnita x_0 cuyo valor ya es conocido (barra *slack* en el problema del Flujo de Potencia). Se puede observar entonces que el sistema de ecuaciones a ser resuelto esta representado por las 13 incógnitas x_1 a x_{13}.

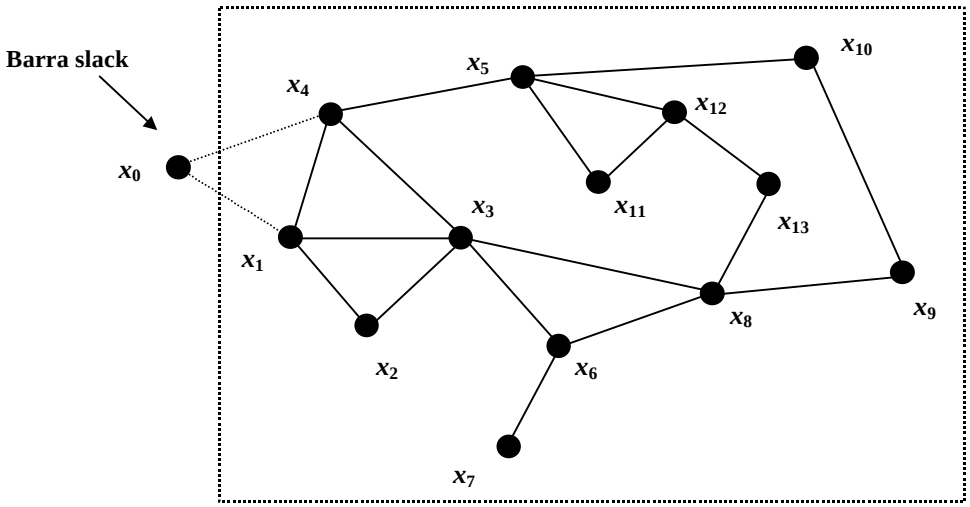


Figura 7.1 : Sistema IEEE-14.

El sistema IEEE-14 se descompuso de todas las formas posibles de manera a resolverlo utilizando las dos workstations descritas con anterioridad. Considerando que la

workstation DEC tiene una capacidad de resolver problemas unas 3 veces mayor que la worstation SUN en el mismo periodo de tiempo, el sistema fue particionado de forma tal que el procesador DEC resuelva 10 barras y el procesador SUN 3 barras. De esta forma, el número de particiones a ser analizado es:

$$ncp = C_{10}^{13} - \frac{13!}{10!\,(13-10)!} - 286$$

Las particiones resultantes fueron numeradas de 1 a 286. Para cada una de las 286 particiones, el problema de Flujo de Potencia fue resuelto en forma paralela síncrona y asíncrona, siendo utilizada la máquina DEC para verificar la tolerancia. Las magnitudes medidas y registradas en la resolución de cada partición fueron las siguientes:

- *Tiempo físico*: es el tiempo de reloj utilizado por el sistema hasta llegar a la solución buscada, medido en el procesador más rápido.
- *Tiempo de CPU de la DEC*: es el tiempo efectivamente utilizado por el procesador de la máquina DEC hasta la solución global del problema, sin considerar el tiempo que dicha máquina ha utilizado en atender otros procesos concurrentes con la resolución del Flujo de Potencia Eléctrica.
- *Número de iteraciones:* es el número de veces que se actualizaron las variables locales de la workstation DEC.

Los valores experimentales así medidos fueron utilizados para elaborar las tablas y gráficos que se muestran en las secciones siguientes.

7.2.1 Particiones generadas

El método de partición fue aplicado al Sistema IEEE-14 utilizando diversas ternas de parámetros *vlim*, *ngrup* y *nvec* en la selección automática de semillas. Así fueron obtenidas dos particiones: la numero 56 y la número 59, cuya numeración se corresponde con el orden establecido al identificar las 286 posibles particiones. Tomando como criterio de selección el parámetro propuesto en este trabajo, es decir, el radio espectral $\rho_b(\mathbf{H})$ de la matriz de comparación, la partición 59 es la seleccionada como óptima.

Por su parte, el método de descomposición más utilizado en la actualidad, la *Descomposición* ε, no pudo organizar dos subsistemas con las dimensiones deseadas, por lo cual no se pudo comparar su desempeño frente al método propuesto.

En las figuras 7.2 y 7.3 se muestran las dos particiones generadas aplicando la metodología presentada en este trabajo.

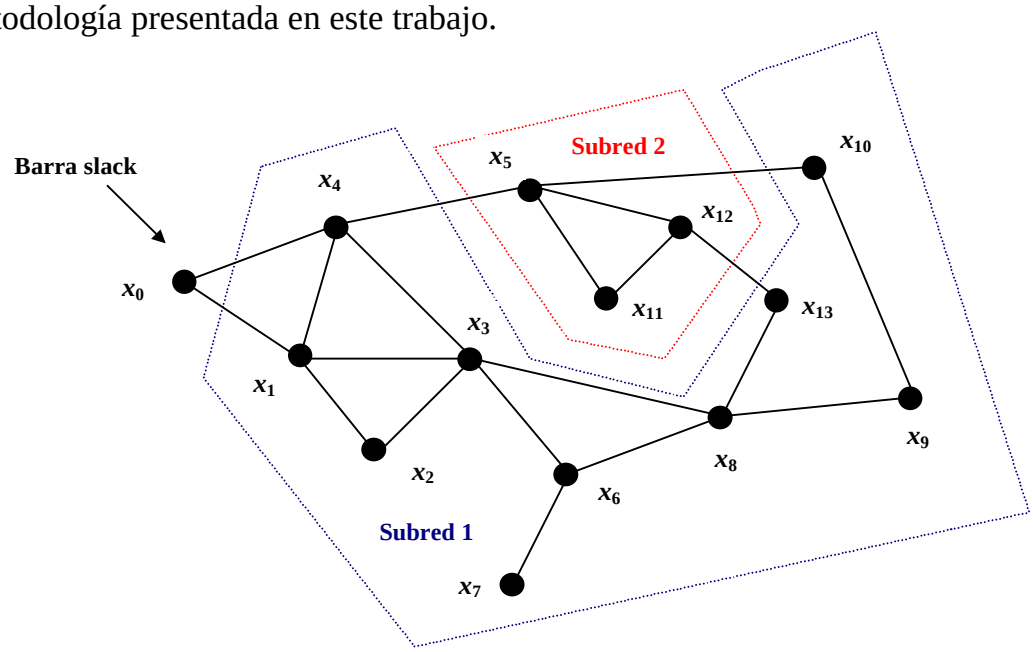


Figura 7.2 : Mejor partición generada por el método propuesto.
(partición 59)

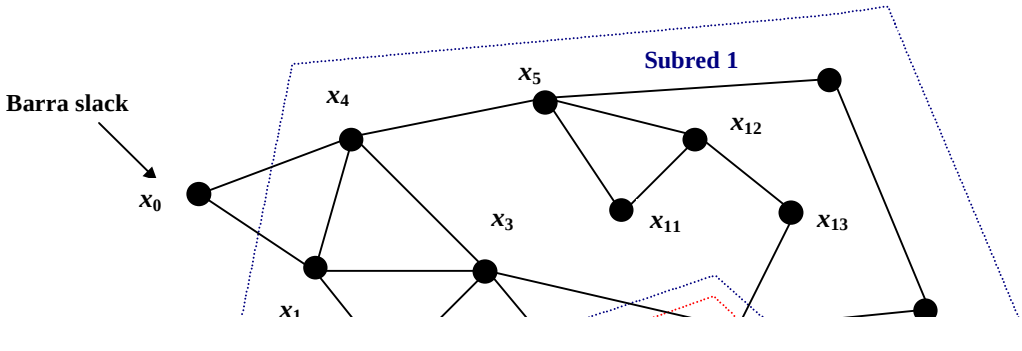


Figura 7.3 : Segunda partición generada el método propuesto.
(partición 56)

7.2.2 Resolución síncrona

En la tabla 7.1 se muestra el posicionamiento relativo de las particiones identificadas más arriba con respecto a la totalidad de particiones posibles.

Resolución síncrona del sistema IEEE-14						
Identificador de Partición	Ubicación[+] respecto a:					
	N° de iteraciones	% sup.	Tiempo Real	% sup.	Tiempo de CPU	% sup.
59	3ra posición	1.04	4ta posición	1.39	3ra posición	1.04
56	47a posición	16.43	34a posición	11.88	43a posición	15.03

[+] Existen 286 posibles particiones

Tabla 7.1: Posición de las particiones generadas en el ranking.
Resolución síncrona del sistema IEEE-14.

En las gráficas 7.4, 7.5 y 7.6 se puede observar el comportamiento de las diferentes particiones con respecto a los valores medidos. Se resaltan especialmente las particiones generadas por el método propuesto:

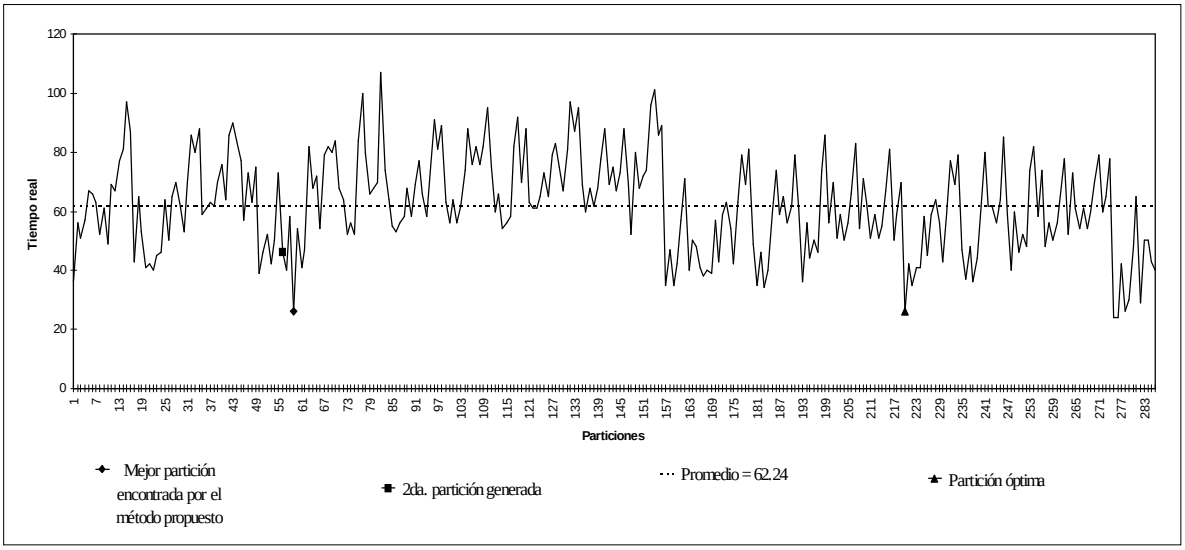


Figura 7.4 : Tiempo real. Resolución síncrona del sistema IEEE-14.

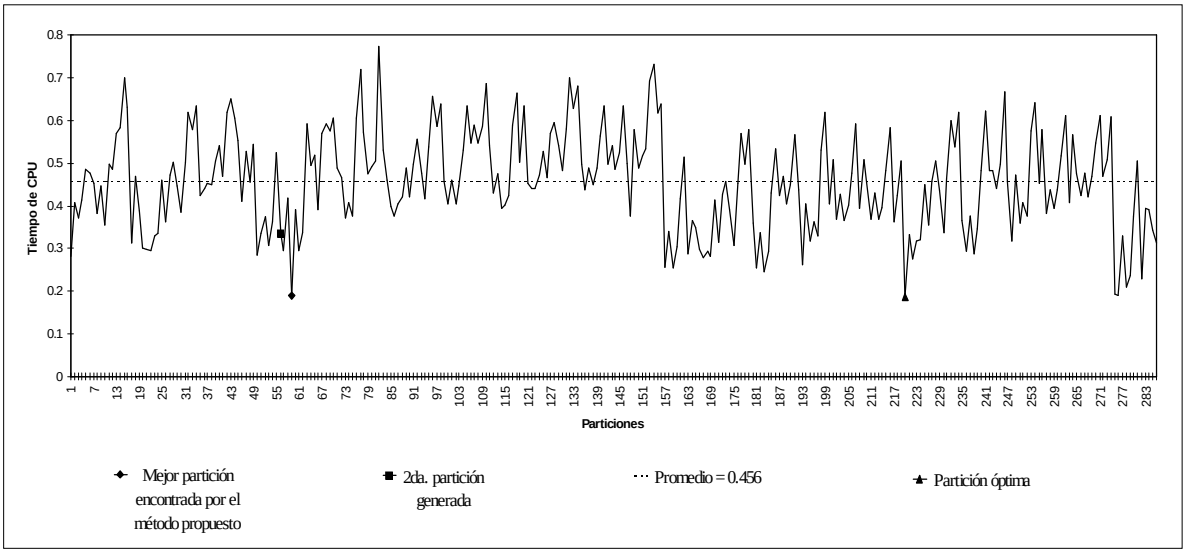


Figura 7.5 : Tiempo de CPU. Resolución síncrona del sistema IEEE-14.

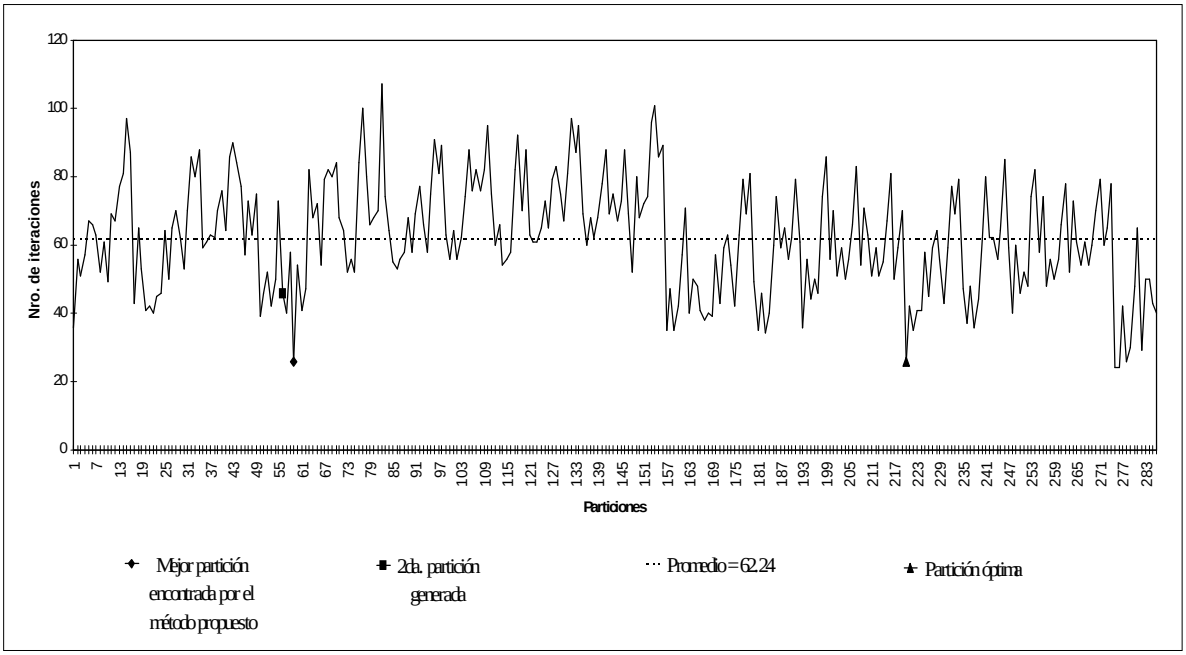


Figura 7.6 : Iteraciones. Resolución síncrona del sistema IEEE-14.

Puede observarse fácilmente que las dos particiones generadas por el método propuesto mostraron un comportamiento altamente satisfactorio, presentando un desempeño muy superior al promedio. La partición indicada como la mejor disponible según el criterio de selección propuesto se encontró por lo general dentro del 1.40 % superior con respecto a todas las magnitudes medidas. Es importante resaltar que ni la *Descomposición* ε (sección 3.2) ni el *Método de la Semilla* de Vale et al. (sección 3.3) consiguieron descomponer el sistema IEEE-14 en la proporción deseada.

7.2.3 Resolución asíncrona

Se resolvieron en forma asíncrona las 286 posibles particiones, es decir, de forma tal que los procesadores hagan uso del valor más actualizado que disponían para cada variable calculada por los otros procesadores del sistema distribuido sin implementar barreras de sincronización (sección. 2.2). Los resultados experimentales arrojados son

ligeramente diferentes a los obtenidos en la resolución síncrona. Si bien con respecto al número de iteraciones el desempeño de las particiones generadas se mostró inferior que antes, hubo un repunte en lo que respeta a los tiempos reales y de CPU, encontrándose inclusive que la mejor partición seleccionada por el método es efectivamente la óptima. Esto puede apreciarse mejor en la tabla 7.2:

Resolución asíncrona del sistema IEEE-14						
Número de Partición	Ubicación[+] respecto a:					
	N° de iteraciones	% sup.	Tiempo Real	% sup.	Tiempo de CPU	% sup.
59	7ª posición	2.44	1ra posición	0.349	1ra posición	0.349
56	82ª posición	28.67	45ª posición	15.73	56ª posición	19.58

[+] Existen 286 posibles particiones

Tabla 7.2: Posición de las particiones generadas en el ranking.
Resolución asíncrona del sistema IEEE-14.

Se muestran a continuación las diversas gráficas donde se aprecia el desempeño de todas las particiones estudiadas en este análisis exhaustivo.

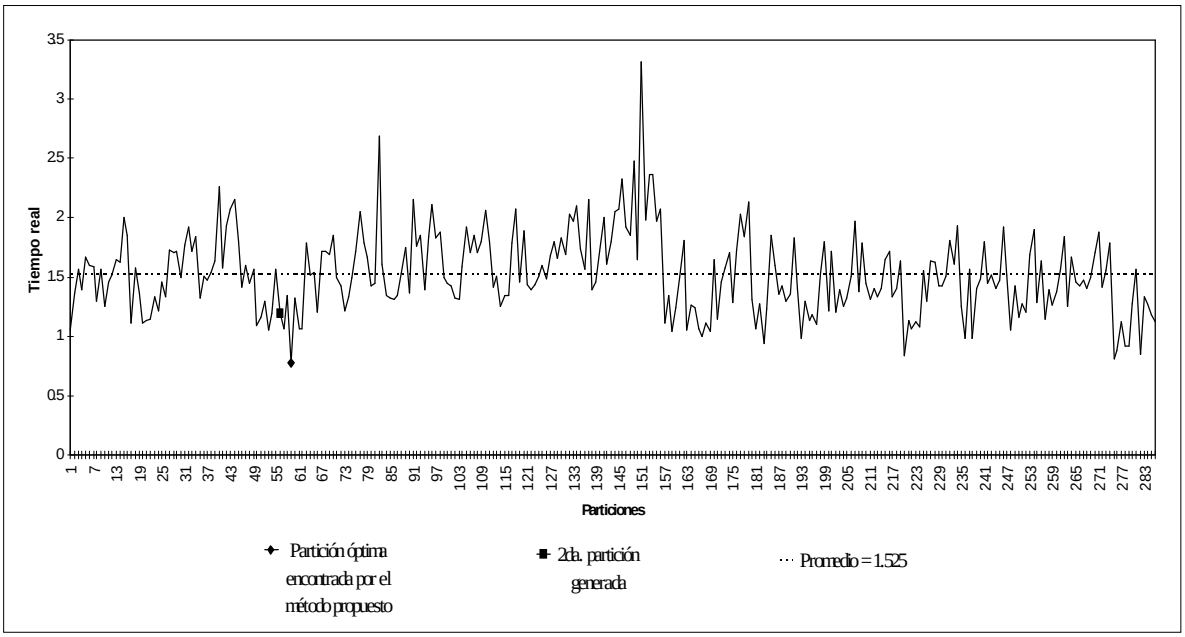


Figura 7.7 : Tiempo real. Resolución asíncrona del sistema IEEE-14.

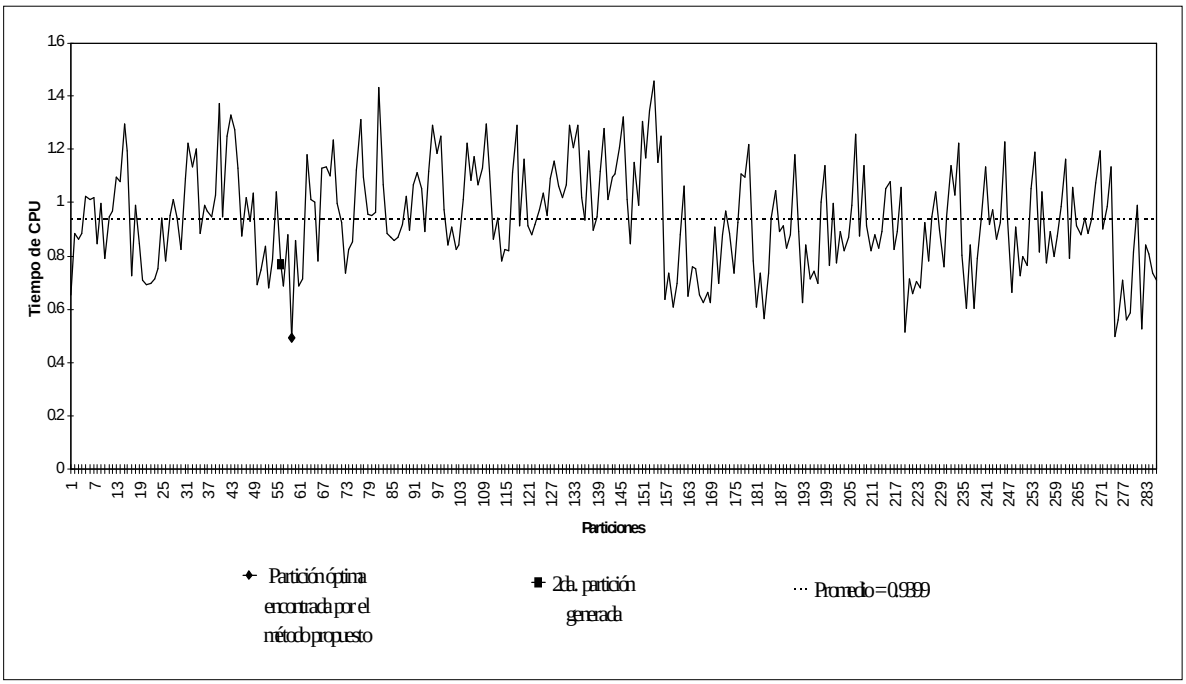


Figura 7.8 : Tiempo de CPU. Resolución asíncrona del sistema IEEE-14.

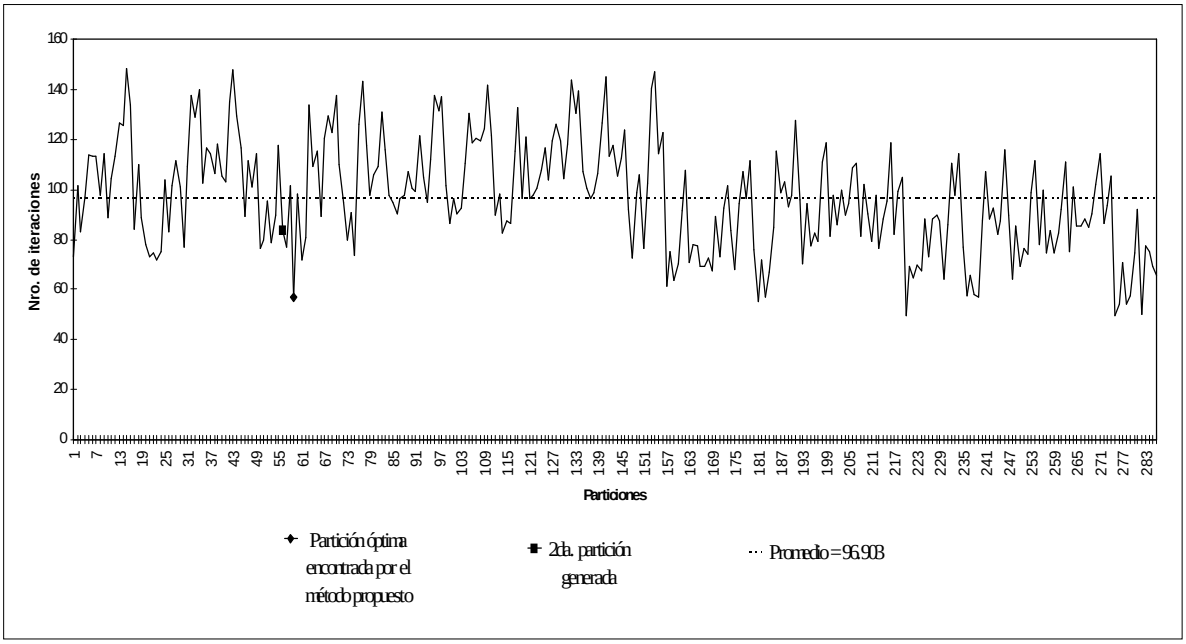


Figura 7.9 : Iteraciones. Resolución asíncrona del sistema IEEE-14.

En conclusión, el método propuesto demostró un excelente desempeño en la solución asíncrona del sistema IEEE-14, detectando a la partición que consigue resolver el problema del flujo de potencia con el menor tiempo de procesamiento. Es importante recordar que es justamente la implementación asíncrona la más interesante computacionalmente, por aprovechar mejor los recursos computacionales al no perder tiempo en barreras de sincronización.

7.2.4 Comportamiento del parámetro de selección propuesto

Una de las motivaciones principales del presente trabajo fue la de proponer un parámetro válido que pudiera servir como una clara referencia de la calidad de una partición en términos del procesamiento paralelo. En las tablas y gráficos que siguen, se analiza el comportamiento del radio espectral de la matriz de comparación como parámetro de selección, comparándolo con el parámetro más utilizado hasta el momento,

propuesto por Vale et al.[30], que consiste en la sumatoria de los valores absolutos de todas las ligaciones cortadas por la partición, o parámetro “Par_A” (sección 5.5).

La tabla 7.3 indica las correlaciones entre las diversas magnitudes medidas y los parámetros de selección, tanto en el caso síncrono como en el asíncrono. Se verificó en ambos casos que el radio espectral $\rho(\mathbf{H})$ presenta una mejor correlación que el parámetro de selección Par_A con respecto a los tiempos de procesamiento y número de iteraciones.

Correlaciones	Iteraciones		Tiempo Real		Tiempo de CPU	
	sinc.	asinc.	sinc.	asinc.	sinc.	asinc.
$\rho(\mathbf{H})$	0.414	0.87	0.378	0.343	0.41	0.392
Par_A	0.0377	-0.05	0.0104	0.089	0.0137	0.0137

Tabla 7.3 : Correlaciones: sistema IEEE-14.

Se muestran además en las figuras 7.10 a 7.13 las gráficas comparativas entre los valores normalizados de los parámetros de selección analizados y los tiempos reales medidos

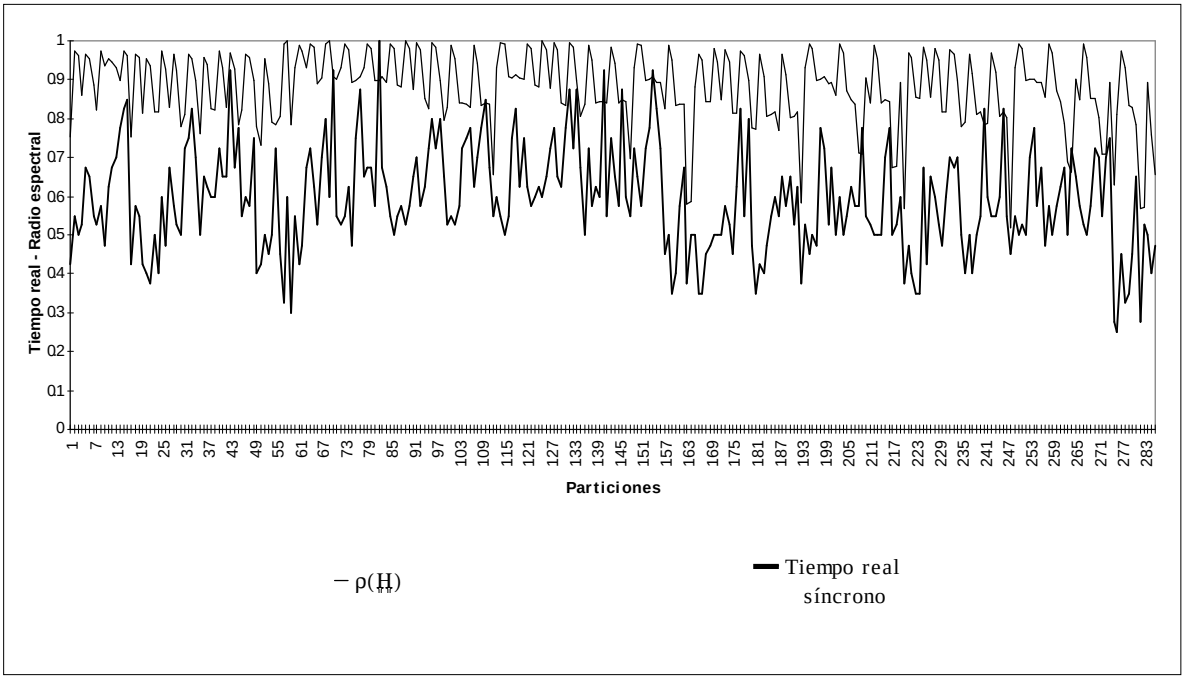


Figura 7.10 : Gráfica normalizada: Tiempo real síncrono - b (**H**).
Sistema IEEE-14.

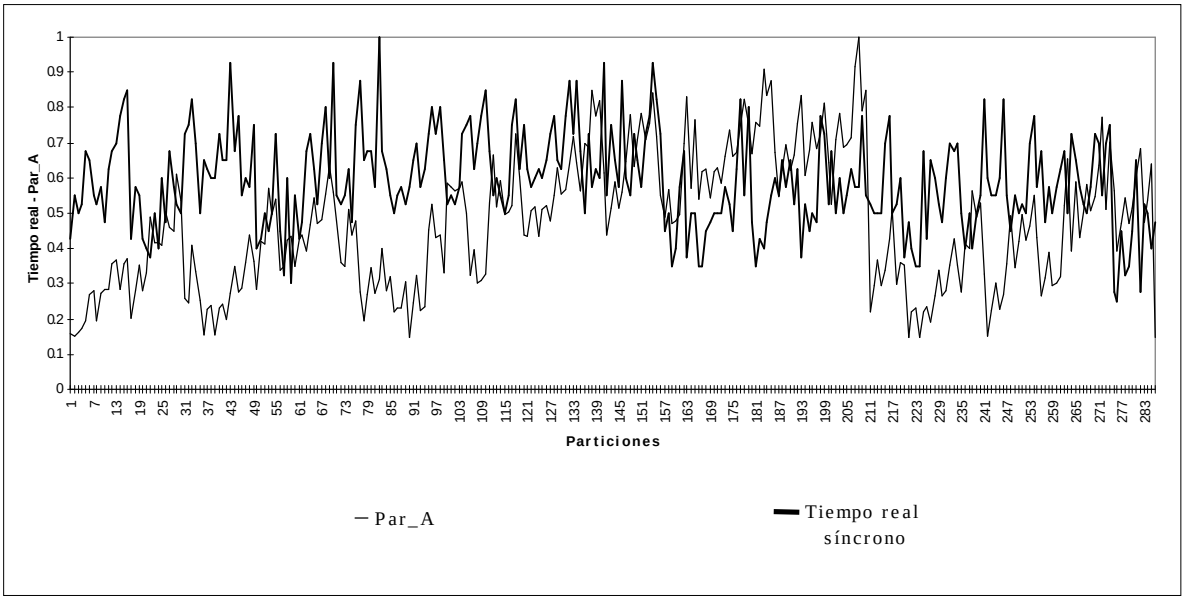


Figura 7.11 : Gráfica normalizada: Tiempo real síncrono - Par_A.
Sistema IEEE-14.

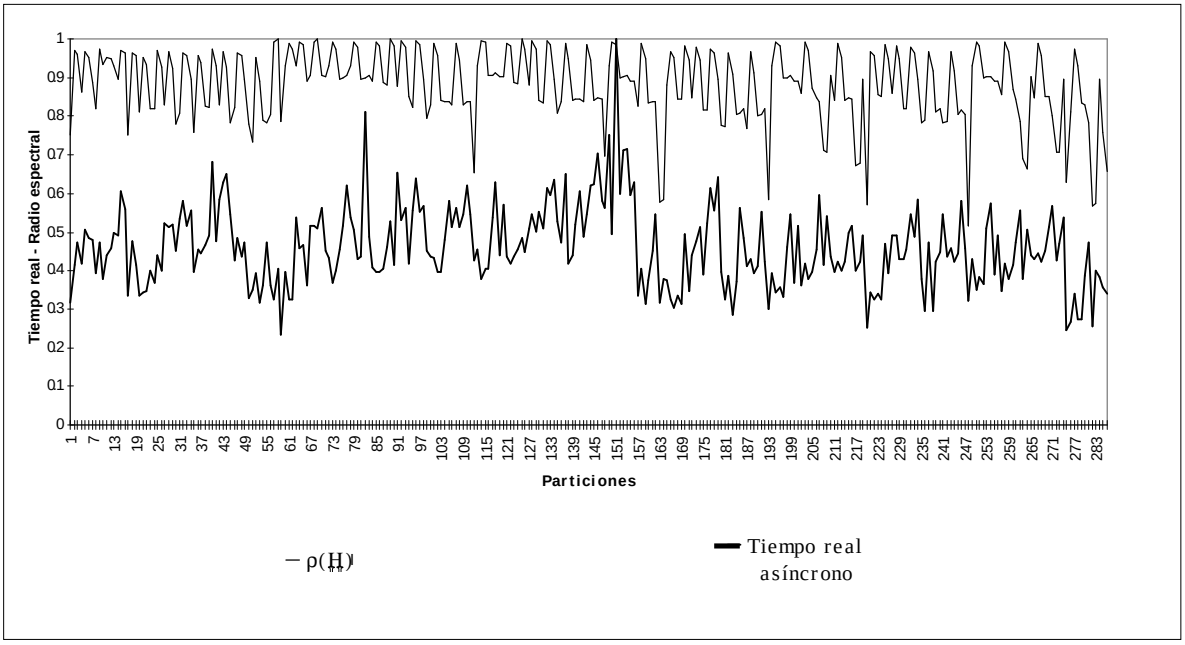


Figura 7.12 : Gráfica normalizada: Tiempo real asíncrono - ρ (**H**).
Sistema IEEE-14.

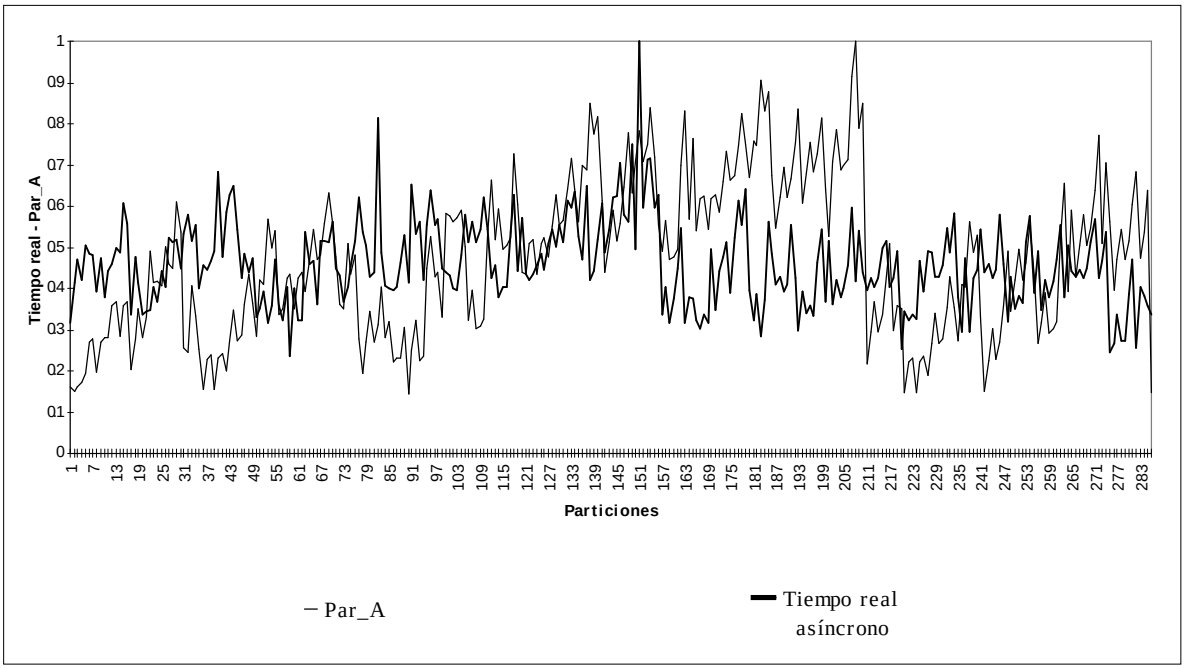


Figura 7.13: Gráfica normalizada: Tiempo real asíncrono - Par_A.
Sistema IEEE-14.

Conforme a estos resultados, el parámetro de selección $\rho(\mathbf{H})$ propuesto en este trabajo es la mejor referencia para determinar la calidad de una dada partición frente a otra. Sin embargo, esto redunda en un mayor costo computacional, pues el cálculo del radio espectral de una matriz es una operación cuya complejidad aumenta en forma exponencial con la dimensión de la matriz de comparación **H**.

7.3 Resolución del sistema IEEE-118

Con la intención de verificar la eficiencia del método propuesto utilizando sistemas de ecuaciones de mayor tamaño, se seleccionó al sistema IEEE-118. Es evidente que un estudio exhaustivo sobre todas las formas posibles de partir este sistema es imposible, debido a la dimensión del problema. Si, por ejemplo, se deseara partirlo simplemente en dos subsistemas iguales, tendríamos la siguiente cantidad de particiones:

$$ncp = C_{58}^{117} = \frac{117!}{58! \times (117 - 58)!} = 1.2157 \times 10^{34}$$

Evaluar dicho número de particiones es del todo impráctico, por la cantidad de trabajo computacional implicado, pues resultaría imposible de experimentar aún durante toda la vida útil de una persona, utilizando las computadoras hoy disponibles.

A raíz de esta imposibilidad práctica, las experimentaciones realizadas sobre este sistema de ecuaciones se remitieron a las siguientes particiones:

- Las particiones generadas por el método propuesto. Dentro de ellas se tendrán en cuenta tanto las generadas utilizando semillas seleccionadas automáticamente como las generadas utilizando semillas asignadas manualmente.
- La partición generada por la *Descomposición* ε.
- Las particiones realizadas manualmente sobre el grafo del sistema de ecuaciones. Estas particiones fueron organizadas tomando en cuenta criterios empíricos y la experiencia

del operador sobre el problema.

- Particiones generadas de forma aleatoria. Se analizaron un centenar de estas particiones.

Nuevamente, el sistema se resolvió con implementaciones síncronas y asíncronas, pero esta vez con 4 procesadores de similar performance, por lo que el problema fue descompuesto en 4 subredes de aproximadamente igual dimensión. En este caso solamente se midió el tiempo real utilizado por el sistema distribuido en llegar a la solución.

La representación del sistema eléctrico se muestra en la figura 7.14.

7.3.1 Particiones generadas

Para aplicar el método propuesto a la descomposición del Sistema IEEE - 118, se hizo necesario un análisis mas detenido de los parámetros utilizados en la selección de semillas, tomándose finalmente valores dictados por criterios empíricos válidos en el marco de este estudio.

Fueron adoptados los siguientes valores de los parámetros:

- ***vlim*** = un valor tal que sean analizadas el 10% del total de incógnitas (para el sistema en estudio *vlim*=4.9 produce 11 candidatas a semillas).
- ***ngrup*** = 15, por ser el mayor valor de *ngrup* para el cual la PC utilizada tiene suficiente memoria.
- ***nvec*** = 10, pues es el número máximo para el cual aún se obtienen las 4 semillas requeridas.

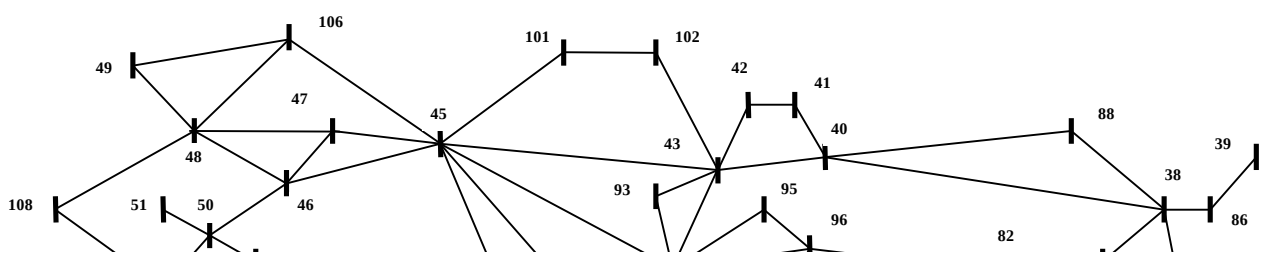


Figura 7.14: Sistema IEEE - 118
OBS: Las barras | representan las incógnitas x_i del sistema de ecuaciones.

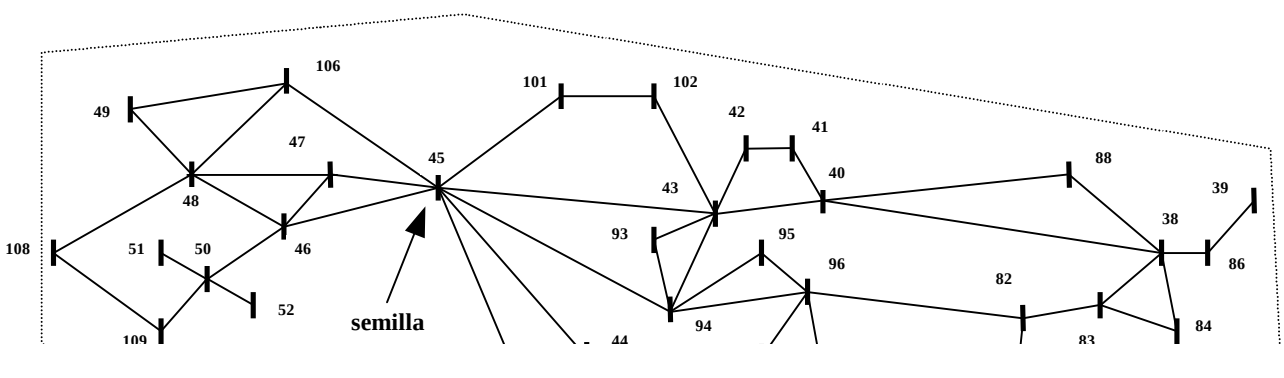


Figura 7.15 : Sistema IEEE - 118. Partición generada por el método propuesto.

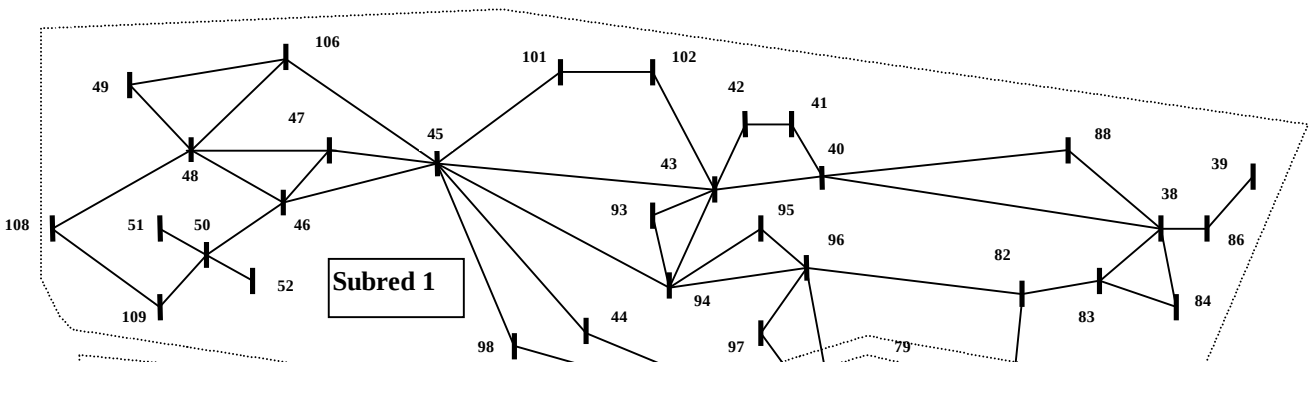


Figura 7.16 : Sistema IEEE - 118. Partición generada por la *Descomposición* $_{\varepsilon}$..

Las semillas seleccionadas en forma automática por el método fueron las incógnitas x_{45}, x_7, x_{22} y x_{85}. Puede observarse fácilmente en la figura 7.14 que dichas incógnitas semillas constituyen efectivamente centros de agrupamientos de incógnitas, lo

que las convierte en candidatas idóneas para una buena partición.

En la figura 7.15 se muestra la partición generada por el método propuesto, mientras que la figura 7.16 muestra la partición generada aplicando la *Descomposición* $_{\varepsilon}$.

7.3.2 Resolución síncrona

El problema del Flujo de Potencia para el sistema IEEE-118 fue resuelto utilizando una amplia variedad de particiones, citadas en la sección 7.4.

A la hora de analizar las particiones generadas por el método propuesto, conviene resaltar aquí que dicho método consta principalmente de 4 procesos: clasificación de incógnitas, selección de semillas, generación de la partición y evaluación de particiones y selección. En la etapa de generación de la partición, puede ser obtenida una descomposición a partir de un grupo cualquiera de semillas, pudiendo estas ser seleccionadas de manera automática o manual. Con el objeto de analizar la conveniencia o no de seleccionar las semillas automáticamente, se estudiaron también 4 particiones obtenidas a partir de semillas asignadas de forma manual, utilizando criterios heurísticos basados en la experiencia del operador. Se analizaron además 3 particiones realizadas con criterios geográficos, utilizando la experiencia de expertos con respecto al sistema eléctrico en estudio.

En la tabla 7.4 se muestra el desempeño de las mejores particiones obtenidas, siendo el tiempo real hasta la resolución del problema la magnitud medida en este caso.

Posición	Tipo de partición	Selecc. de semillas	Tiempo (seg.)
1ª posición	Generada por el método propuesto	manual	57
2ª posición	Generada por el método propuesto	manual	58
3ª posición	Generada por el método propuesto	manual	62
3ª posición	Generada por la *Descomposición* $_{\varepsilon}$	---	62
4ª posición	Generada por el método propuesto	manual	87

5ª posición	Generada por el método propuesto	automática	95
6ª posición	Partición manual (geográfica)	---	139
7ª posición	Partición manual (geográfica)	---	149
8ª posición	Partición manual (geográfica)	---	289

Tabla 7.4 : Desempeño de las particiones estudiadas en la resolución síncrona.
Sistema IEEE-118.

Puede verificarse que casi la totalidad de las particiones que se encontraron en los mejores puestos fueron generadas por el método propuesto, aunque utilizando selección manual de semillas. La partición generada utilizando la selección automática de semillas no tuvo un desempeño tan satisfactorio como estas últimas, ubicándose sin embargo en una mejor posición que las particiones obtenidas manualmente por un especialista.

La partición generada aplicando la *Descomposición* ε tuvo un comportamiento bastante satisfactorio, aunque inferior por lo general al presentado por las particiones generadas por el método propuesto.

Adicionalmente a las particiones analizadas más arriba, se resolvió el sistema utilizando 100 particiones generadas en forma aleatoria. Es un hecho notable el que ninguna de esas 100 particiones pudo converger a la solución, demostrando así una vez más que la forma en que la red es descompuesta influye de manera decisiva en el comportamiento de los métodos de resolución.

7.3.3 Resolución asíncrona

Al utilizar el asincronismo en la resolución de problemas, los procesadores son utilizados de una manera más eficiente, ya que no existen “tiempos muertos”, es decir, intervalos de tiempo en los cuales los procesadores se encuentran detenidos esperando resultados de los demás procesadores. La dificultad consiste en encontrar una partición lo suficientemente buena como para asegurar la convergencia y aprovechar los menores tiempos de procesamiento, siendo que el comportamiento de los métodos de resolución bloque-iterativos presenta características que hacen que la convergencia de los mismos

sea difícil de obtener en un contexto asíncrono [4, 8].

De lo anterior concluimos que los resultados obtenidos resolviendo en forma asíncrona el Sistema IEEE-118 son más importantes en el contexto de la computación distribuida que los obtenidos en la resolución síncrona. En la tabla 7.5 se muestra el comportamiento de las diversas particiones estudiadas.

Posición	**Tipo de partición**	**Selecc. de semillas**	**Tiempo (seg.)**
1ra posición	Generada por el método propuesto	automática	30
2da posición	Generada por el método propuesto	manual	31
3ra posición	Generada por el método propuesto	manual	43
4ta posición	Generada por el método propuesto	manual	44
5ta posición	Generada por la *Descomposición* $_{\varepsilon}$	---	189
6ta posición	Generada por el método propuesto	manual	no converge.
6ta posición	Partición manual (geográfica)	---	no converge
6ta posición	Partición manual (geográfica)	---	no converge
7a posición	Partición manual (geográfica)	---	no converge

Tabla 7.5 : Desempeño de las particiones estudiadas en la resolución asíncrona.
Sistema IEEE-118.

Es evidente aquí que el método propuesto fue capaz de generar las particiones más idóneas para la resolución asíncrona. La selección automática de semillas demostró en esta oportunidad ser la mejor de las opciones encontradas experimentalmente, recordando la imposibilidad de hacer un estudio exhaustivo de todas las alternativas.

La partición obtenida aplicando la *Descomposición* $_{\varepsilon}$ tuvo un desempeño muy inferior a la mayoría de las particiones generadas por el método propuesto, necesitando un tiempo de resolución 6 veces mayor que la partición encontrada por el método propuesto.

Nótese finalmente que las particiones generadas por el especialista no convergen, por lo que resulta evidente la utilidad de los métodos automáticos, motivo de estudio del presente trabajo.

CAPITULO 8: CONCLUSIONES

Dada la necesidad de estudiar sistemas de ecuaciones cada vez más grandes y complejos, resulta evidente la importancia de poseer herramientas computacionales capaces de estudiar soluciones más eficientes y apropiadas a los sistemas computacionales existentes.

Ante esta situación, la computación distribuida emerge como una herramienta altamente viable en la realidad de nuestro país por su capacidad de ir resolviendo problemas cada vez mayores aprovechando la capacidad computacional existente en forma de redes de computadoras, las cuales podrán ir creciendo conforme con las nuevas exigencias.

Sin embargo, el óptimo aprovechamiento de los sistemas distribuidos se encuentra limitado por la capacidad de descomponer los sistemas a ser estudiados en subsistemas menores capaces de ser resueltos eficientemente con los sistemas distribuidos existentes. Este es el punto crucial de los estudios presentados en este trabajo.

En efecto, se presentó un método capaz de partir un sistema de ecuaciones en subsistemas menores de proporciones deseadas y con características matemáticas aceptables para su resolución en un contexto distribuido. Es más, resultados experimentales con el sistema IEEE-118 demostraron que el método propuesto es superior a otros similares ya publicados hasta la fecha, cuando es implementado en un contexto asíncrono (tabla 7.5, sección 7.3.3).

Entre las características que hacen al método propuesto una excelente alternativa frente a otros métodos de descomposición utilizados hasta la fecha, podemos resaltar las que siguen:

- Permite obtener descomposiciones con tamaños relativos deseados por el operador, conforme sea el sistema distribuido a ser utilizado en la resolución del problema correspondiente. En este sentido, se recuerda que la *Descomposición* ε no tiene control sobre el tamaño relativo de los subsistemas generados, mientras que el *Método de la Semilla* solo genera particiones de igual dimensión. Se enfatiza la importancia de esta característica para el óptimo aprovechamiento de las redes de computadoras con diferentes *performances* relativas o redes heterogéneas.

- La robustez del método en la selección de los diversos parámetros de partición, como por ejemplo, las semillas utilizadas. En efecto, en los estudios experimentales para la descomposición del sistema IEEE-118, quedó claramente demostrado que aún cambiando las semillas se generan las mismas buenas particiones debido a que, aunque las semillas en sí no sean los principales centros de agrupamientos de incógnitas, las mejores candidatas a semillas son anexadas entre las primeras incógnitas agrupadas de la descomposición en formación, pasando a dominar la dinámica de anexación de barras, y generando por consiguiente la misma partición que la generada con una buena selección de semillas.

- La capacidad de elegir o de seleccionar la mejor de varias particiones posibles utilizando un parámetro de selección matemáticamente fundamentado como lo es $_{b}$ (**H***)* en lugar de parámetros meramente intuitivos presentados en [30]. En efecto, en la sección 7.2.4 quedó demostrado que el parámetro de selección aquí propuesto tiene una mejor correlación con las magnitudes cuya optimización se desea, y de hecho, en el ejemplo IEEE-14 el parámetro Par_A no hubiera elegido la mejor partición, mientras que $_{b}$ (**H***)* si lo hace.

- El método propuesto es lo suficientemente general como para poder utilizarlo en la resolución de diversos problemas de ingeniería u otras áreas que conlleven la resolución de sistema de ecuaciones de gran porte.

En cuanto a los resultados experimentales, se pudo comprobar que el método propuesto genera mejores particiones que las generadas en promedio con un criterio aleatorio, siendo en general una muy buena partición e inclusive la optima teórica en algunos casos, como fuera presentado en la sección 7.2.3 para el sistema IEEE-14. Al mismo tiempo pudo comprobarse experimentalmente que las particiones generadas resultaron superiores a las generadas por la *Descomposición* ε.

En resumen, el presente trabajo brinda una solución superior a las hoy existentes para la descomposición de sistemas de ecuaciones en subsistemas menores y suficientemente desacoplados como para permitir su resolución utilizando sistemas distribuidos heterogéneos, inclusive en un contexto asíncrono, lo que permitiría que países como el nuestro aprovechen su capacidad computacional instalada en el estudio y búsqueda de solución de los grandes problemas de ingeniería que puedan ir presentándose.

APÉNDICES

APÉNDICE A:

CÓDIGO EN LENGUAJE C++ PARA LAS ETAPAS DEL MÉTODO PROPUESTO

CÓDIGO EN LENGUAJE C++ DE LAS ETAPAS DEL MÉTODO PROPUESTO

Debido a la extensión del programa, en este apéndice se presentan solo los códigos en lenguaje C++ para las etapas del método propuesto y no los detalles de la interface con el usuario y otros procesos o funciones internas. La versión completa en medio magnético está disponible en disquete adjunto a la presente.

```
/* ************************************************************** */
/* Etapa1 :                                                       */
/* Función que lee la matriz M de un archivo, la carga en la matriz */
/* Mcoef[][] y calcula el valor de los pesos de las incógnitas    */
/* -------------------------------------------------------------- */
/* Entradas: nombre del archivo                                   */
/* Salidas: matriz de coeficientes, Mcoef[][] y                   */
/*          vector de pesos, Pesos                                */
/* -------------------------------------------------------------- */
void Etapa1(char *arg1)
{
FILE *arch1;
int i,j;
float max,        /* variable auxiliar en el ordenamiento de pesos     */
  bs,
  ep,
  M ;             /* sumatoria del valor absoluto de las ligaciones    */

int  nl,          /* contador del número de ligaciones */
     cont,        /* contador del número de pesos ordenados  */
     pos;         /* variable auxiliar en el ordenamiento de los pesos */

/* apertura del archivo de la matriz M del sistema de ecuaciones       */
arch1=fopen(arg1,"r")

nl=0;             /* contador del número de ligaciones                 */
M=0;              /* sumatoria del valor absoluto de las ligaciones    */

/* cargar la matriz de coeficientes Mcoef                              */
for(i=1;i<=NN;i++)
  for(j=1;j<=NN;j++)
    {
    fscanf(arch1,"%f",(Mcoef+(NN+1)*i+j));
      if(i!=j && *(Mcoef+(NN+1)*i+j)!=0)
        {
        nl++;    /* se incrementa el contador de número de ligaciones */
        M=M+fabs(*(Mcoef+(NN+1)*i+j));
        }
    }
```

```
M=M/nl;

/* inicialización a 0 de los pesos                                    */
for(i=1;i<=NN;i++)
   *(Peso+i)=0;

for(i=1;i<=NN;i++)
   {
   for(j=1; j<=NN; j++)
      {
      if(i==j)
         continue;

      if(*(Mcoef+(NN+1)*i+j)==0)
         continue;

      bs=*(Mcoef+(NN+1)*i+j)/(*(Mcoef+(NN+1)*i+i));
      ep=*(Mcoef+(NN+1)*i+j)/(*(Mcoef+(NN+1)*i+i)*M);
      *(Peso+i)=*(Peso+i)+pow((double)fabs(bs),(double)fabs(ep));
      }

   *(Tem1+i)=*(Peso+i);
   }

cont=0;
while(cont!=NN)
   {
   max=0;
   for(i=1;i<=NN;i++)
      {
      if(*(Tem1+i)>max)
         {
         max=*(Tem1+i);
         pos=i;
         }
      }
   cont++;
   *(Tem1+pos)=0;
   *(Posi+cont)=pos;
   }
fclose(arch1);
}

/* ************************ fin de Etapa1 ************************** */
```

```
/* ***************************************************************** */
/* Etapa2                                                            */
/* Función que selecciona las semillas a ser utilizadas en el        */
/* algoritmo de partición                                            */
/* Parámetros : np    : número de procesadores del sist. distribuido */
/*              psup  :                                              */
/*              ngrup : indica cuantas incógnitas deben ser agrupadas */
/*                      alrededor de cada candidata a semilla para   */
/*                      luego realizando la sumatoria de sus pesos,  */
/*                      decidir cuales serán la np semillas a ser    */
/*                      utilizadas                                   */
/*              nvec  : se utiliza para evitar que las incógnitas    */
/*                      fuertemente acopladas entre si sean semillas */
/*                      al mismo tiempo                              */
/* ----------------------------------------------------------------- */
```

```
void Etapa2(int np, float psup, int ngrup, int nvec)
{
float maxs;   /* variable auxiliar para calcular la máxima sumatoria   */

int nk,       /* número de incógnitas a ser evaluadas                  */
    i,j,      /* variables de ciclo                                    */
    nsem,     /* número de semillas seleccionadas en un momento dado   */
    pos,      /* variable auxiliar para calcular la máxima sumatoria   */
    ninc;     /* cantidad de incógnitas agrupadas en un momento dado   */

nk=(int)(NN*psup/100);  /* se calcula el nro. de elem. del conjunto K  */

for(i=1;i<=nk;i++)   /* se incluyen en K las nk incógnitas mas pesadas */
  *(K+i)=*(Posi+i);

for(i=1;i<=nk;i++)   /* para cada elemento de K                       */
  {
  for(j=1;j<=NN;j++)
    /* inicializa el vector de disponibilidad de incógnitas           */
    *(Liber+j)=0;
  *(Agrup+(NN+1)*i+1)=*(K+i)
  *(Liber+(*(K+i)))=1;       /* la elimina como disponible            */
  ninc=1;                    /* hay una incógnita en el subconjunto   */
  while(ninc<=ngrup)
    {
    /* selecciona la incógnita mas pesada entre las adyacentes        */
    *(Agrup+(NN+1)*i+ninc+1)=Pesomax(Agrup,i,ninc);
    /* la elimina como disponible                                     */
    *(Liber+(*(Agrup+(NN+1)*i+ninc+1)))=1;
    ninc++;
    }
}

/* inicialización de las sumatorias a cero                           */
for(i=1;i<=nk;i++)
    *(Sum+i)=0;

/* calculo de las sumatorias de las incógnitas agrupadas             */
for(i=1;i<=nk;i++)
  for(j=1;j<=ngrup;j++)       /* sumar las ngrup incógnitas agrupadas */
    *(Sum+i)=*(Sum+i)+(*(Peso+*(Agrup+(NN+1)*i+j)));
```

```
nsem=0;

/* mientras no se seleccionen la cantidad de semillas deseada          */
while(nsem<np)
   {
   maxs=0;
   for(i=1;i<=nk;i++)         /* para cada valor de sumatoria          */
     /* si el valor de sumat.es mayor es el mayor hasta el momento     */
     if(*(Sum+i)>maxs)
        {
        maxs=*(Sum+i);        /* seleccionarlo como máximo             */
        pos=i;                /* guardar su identificación             */
        }

   /* si la condición de nvec se cumple                                */
   if(Vernviz(Agrup,Sem,nsem,nvec,*(Agrup+(NN+1)*pos+1),nk))
     {
     nsem++; /* se incrementa en 1 el número de semillas seleccionadas */
     /* se incluye a la seleccionada en el vector de semillas          */
     *(Sem+nsem)=*(Agrup+(NN+1)*pos+1);
     /* se elimina a la seleccionada de entre las candidatas           */
     *(Sum+pos)=0;
     }
   /* si la condición de nvec no se verifica                           */
   else
   /* se elimina a la seleccionada de entre las candidatas             */
     *(Sum+pos)=0;
   } /* del while(nsem<np)                                            */

}/* de la Etapa2*/

/* ************************** fin de Etapa2 ************************** */


/* ********************************************************************* */
/* Pesomax                                                               */
/* Función que determina la incógnita con mayor peso de entre las        */
/* adyacentes a un conjunto dado                                         */
/* --------------------------------------------------------------------- */
/* Parámetros:*mat : matriz de agrupamiento                              */
/*             p   : fila a ser analizada de la matriz                   */
/*             n   : incógnitas agrupadas en esa fila                    */
/* --------------------------------------------------------------------- */
int Pesomax(int *mat, int p, int n)
{
float maxp;       /* peso de mayor valor                                 */
int selec,        /* identificación de incógnita                         */
    i,j;          /* variables de ciclo                                  */

maxp=0;
for(i=1;i<=n;i++)       /* para cada elemento del subconjunto            */
   {
   for(j=1;j<=NN;j++)   /* se analiza cada incógnita del sistema         */
     {
     if(*(Liber+j)==1)
       continue;/* si la incog. ya fue incluida, se la salta            */
     /* si la incog.es ady. y su peso es mayor al máximo del momento     */
     if((*(Mcoef+(NN+1)*(*(mat+(NN+1)*p+i))+j)!=0  ||
       *(Mcoef+(NN+1)*j+(*(mat+(NN+1)*p+i)))!=0) && (*(Peso+j)>maxp))
```

```
      {
      maxp=*(Peso+j);  /* se la selecciona como ls de máximo peso     */
      selec=j;         /* se guarda que incógnita es                  */
      }
    }
  }
return(selec);
}
/* ********************** fin de Pesomax ***************************** */

/* ******************************************************************** */
/* Vernviz                                                              */
/* Función que verifica la condición de nvec                            */
/* -------------------------------------------------------------------- */
/* Parámetros:*Agrup : matriz de agrupamiento                           */
/*            *Sem   : vector de semillas ya seleccionadas              */
/*            nsem   : cantidad de semillas ya seleccionadas            */
/*            nvec   : número de la condición de vecindad               */
/*            pos    : incógnita a ser evaluada                         */
/*            nk     : número de incógnitas a ser evaluadas como sem.   */
/* -------------------------------------------------------------------- */
int Vernviz(int *Agrup,int *Sem,int nsem,int nvec,int pos,int nk)
{
int resp,   /* indica si la condición de nvec se cumple o no           */
    i,j,k; /* variables de bucles                                     */

resp=1;
/*para cada una de las semillas seleccionadas previamente              */
for(i=1;i<=nsem;i++)
  {
  for(j=1;j<=nk;j++) /* se detecta la posición de la semilla anterior  */
    {
    /* ubicada la semilla anterior en la posición j                   */
    if(*(Agrup+(NN+1)*j+1)==*(Sem+i))
      {
      /* para cada una de las incógnitas agrupadas                    */
      for(k=2;k<=nvec+1;k++)
        {
        /* verificar si coinciden las incógnitas                      */
        if(pos==*(Agrup+(NN+1)*j+k))
          resp=0;
        }
      }
    }
  }
return(resp);
}
/* ************************ fin de Vernviz ************************** */
```

```
/* ************************************************************** */
/* Etapa3                                                         */
/* Formación de una partición                                     */
/* -------------------------------------------------------------- */
/* Parámetros :   NP      :   Número de procesadores              */
/*                ops     :   opción de solapamiento              */
/*                Limover :   valor de peso para solapamiento parcial  */
/*                nparti  :   número de partición en caso de elegir    */
/*                            selección automática de semillas    */
/* -------------------------------------------------------------- */
```

```
void Etapa3(int NP, int ops, float Limover,int nparti)
{
int i,j,          /* variables de ciclo                           */
     cantproc,    /* contador auxiliar                            */
     ganador,     /* subconjunto ganador                          */
     nagrup,      /* incógnitas agrupadas en un momento dado      */
     unit,        /* variable para controlar el balanceamiento    */
     *Procpe,     /* procesadores que pelean por una candidata    */
     *Nprocpe,    /* nro. de procesadores que pelean por una incógnita */
     *Macop,      /* vector de máximos acoplamientos              */
     *Cand,       /* vector de incog. candidatas de los subconjuntos   */
     *Nady,       /* vector de número de adyacencias              */
     *Pelea,      /* vector indicador de participación en la pelea     */
     *Magrup,     /* matriz de agrupación                         */
     *C;          /* vector de cantidad de incógnitas asociadas   */

char sal[20];     /* variable auxiliar para getchar               */

FILE *salida;     /* archivo que contendrá la partición           */

float *Q,         /* vector de cupo parcial                       */
     maxcop,      /* máximo acoplamiento                          */
     igual;       /* el mayor cupo parcial                        */

/* asignaciones dinámicas de memoria                              */
/* -------------------------------------------------------------- */
/* matriz de agrupacion                                           */
Magrup=new int[(NN+1)*(op7+1)];

/* vector de cantidad de incógnitas asociadas                     */
C=new int[NP+1];

/* vector de cupo parcial                                         */
Q=new float[NP+1];

/* vector de número de adyacencias                                */
Nady=new int[NP+1];

/* vector indicador de participación en la pelea                  */
Pelea=new int[NP+1];

/* vector de incógnitas candidatas de los subconjuntos            */
Cand=new int[NP+1];

/* vector de máximos acoplamientos                                */
Macop=new int[NP+1];

/* procesadores que pelean por una candidata determinada          */
```

```
Procpe=new int[(NP+1)*(NP+1)];

/* número de procesadores que pelean por una incógnita                  */
Nprocpe=new int[NP+1];

unit = 0;

/* inicialización del vector de disponibilidad Liber                    */
/* -------------------------------------------------------------------- */
for(i=1;i<=NN;i++)
   *(Liber+i)=0;    /* todas las incógnitas están disponibles          */

/* cargar las semillas en Magrup                                        */
/* -------------------------------------------------------------------- */
for(i=1;i<=NP;i++)
   *(Magrup+(NN+1)*i+1)=*(Sem+i);

/* eliminación de las semillas como disponibles                         */
/* e inicialización de los vectores C y Q                               */
/* -------------------------------------------------------------------- */
for(i=1;i<=NP;i++)           /* para cada semilla de la partición      */
   {
   *(Liber+(*(Sem+i)))=1;    /* eliminarla como disponible             */
   *(Peso+(*(Sem+i)))=0;
   *(C+i)=1;   /* cada subconjunto tiene una incógnita: la semilla     */
   /* valor inicial de cupo parcial                                     */
   *(Q+i)=(float)(*(C+i))/((float)*(W+i));
   }

/* calculo de la cantidad de adyacentes de cada subconjunto             */
for(i=1;i<=NP;i++)
  *(Nady+i)=cantad(Magrup,i,*(C+i));

nagrup=NP; /* las semillas son las primeras incógnitas agrupadas       */

while(nagrup<NN)       /* mientras haya incógnitas por agrupar          */
  {
  /* actualización del valor de cupo parcial                            */
  for(i=1;i<=NP;i++)
    *(Q+i)=(float)(*(C+i))/((float)*(W+i));  /* valor de cupo parcial */

  /* calculo de la cantidad de adyacentes de cada subconjunto           */
  for(i=1;i<=NP;i++)
    {
    *(Nady+i)=cantad(Magrup,i,*(C+i));
    if(*(Nady+i)==0)
      *(Q+i)=-1;
    }

  /* determinación de cuales subgrupos pelean                           */
  /* ------------------------------------------------------------------ */
  /* se verifica si todos los cupos parciales son iguales               */
  igual=verigual(NP,Q);

  if(igual<0)    /* si todos los cupos parciales son iguales           */
    {
    for(i=1;i<=NP;i++)       /* para cada subgrupo                    */
      if(*(Nady+i)>0)        /* si tiene incógnitas adyacentes        */
        *(Pelea+i)=1;        /* el grupo pelea                        */
      else
```

```
      *(Pelea+i)=0;       /* en caso contrario no pelea            */
  unit = unit + 1;
  }

else           /* en caso contrario                                 */
  {
  for(i=1;i<=NP;i++)            /* para cada procesador            */
    {
    if (*(Q+i)!=-1)             /* si no tiene más adyacentes      */
      *(Pelea+i)=0;             /* no pelea                        */
    else                        /* en caso contrario               */
      if (*(Q+i) < unid + 1)    /* si no alcanzo su siguiente cupo */
        *(Pelea+i)=1;           /* pelea                           */
      else
        *(Pelea+i)=0;           /* no pelea                        */
    }
  }


/* selección de las incog. candidatas de cada subconjunto que pelea */
for(i=1;i<=NP;i++)          /* para cada subconjunto               */
  if(*(Pelea+i)==0)         /* si el subconjunto no pelea          */
    *(Cand+i)=-1;           /* no tiene candidato                  */
  else                      /* de lo contrario                     */
    *(Cand+i)=Pesomax(Magrup,i,*(C+i));

/* calculo de los máximos acoplamientos con las candidatas         */
for(i=1;i<=NP;i++)
  *(Macop+i)=maxacop( Magrup,i,*(C+i),*(Cand+i));

/* se carga la matriz Procpe y el vector Nprocpe                   */
for(i=1;i<=NP;i++)              /* para cada incógnita candidata   */
  {
  cantproc=0;  /* se inicializa el contador del nro de procesadores */
  for(j=1;j<=NP;j++)            /* para cada incógnita             */
    if(*(Cand+j)==*(Cand+i))
      {
      cantproc++;
      *(Procpe+(NP+1)*i+cantproc)=j;
      }
    /* se asigna un valor a la componente de Nprocpe               */
  *(Nprocpe+i)=cantproc;
  }

for(i=1;i<=NP;i++)          /* para cada incógnita candidata       */
  {
  /* si el subconjunto no pelea, se salta al siguiente ciclo       */
  if(*(Cand+i)==-1)
    continue;
  /* si la candidata ya fue seleccionada */
  if(*(Liber+(*(Cand+i)))==1)
    continue;

  igual=1;           /* se inicializa igual                        */

  /* se verifica si todas los acoplamientos son iguales            */
  for(j=2;j<=*(Nprocpe+i);j++)
    {
    /* si hay alguna desigualdad */
    if(*(Macop+(*(Procpe+(NP+1)*i+j)))!=*(Macop+(*(Procpe+(NP+1)*i+1))))
```

```
   igual=-1; /* no son todos los acoplamientos iguales             */
   }

if(igual==-1)  /* si no todos los acoplamientos son iguales       */
   {
   ganador=-1;
   maxcop=0;

   /* se halla con cual subconjunto se halla mas acoplada         */
   for(j=1; j<=*(Nprocpe+i); j++)
      {
      if(*(Macop+(*(Procpe+(NP+1)*i+j)))>maxcop)
         {
         /* si el acoplamiento es mayor al máximo                 */
         maxcop=*(Macop+(*(Procpe+(NP+1)*i+j)));
         /* se guarda cual es el ganador                          */
         ganador=j;
         }
      }
   /* se incluye a la incógnita en el subconjunto ganador         */
   *(Magrup+ (NN+1)*(*(Procpe+(NP+1)*i+ganador))+
    (*(C+*(Procpe+(NP+1)*i+ganador)))+1)=*(Cand+i);
   nagrup++;
   /* el subconjunto ganador tiene una incógnita mas              */
   *(C+(*(Procpe+(NP+1)*i+ganador)))=
          *(C+(*(Procpe+(NP+1)*i+ganador)))+1;
   /* se elimina a la incógnita como disponible                   */
   *(Liber+(*(Cand+i)))=1;

   *(Peso+(*(Cand+i)))=0;
   for(j=1;j<=*(Nprocpe+i);j++)
      {
      /* se sacan de combate a los demás subconjuntos perdedores  */
      *(Cand+(*(Procpe+(NP+1)*i+j)))=-1;
      }
   }

else         /* si todos los acoplamientos son iguales            */
   {
   /* si el peso de la candidata en disputa es mayor que Limover  */
   if((*(Peso+(*(Cand+i)))>Limover)
        && (ops==1))      /* y si se realizara overlapping        */
      {
      /* para cada procesador que la selecciono como candidata    */
      for(j=1;j<=*(Nprocpe+i);j++)
         {
         /* se la incluye en todos los subconjuntos               */
         *(Magrup + (NN+1)*( *(Procpe + (NP+1)*i+j ) ) +
               *(C+(*(Procpe+(NP+1)*i+j)))+1)=*(Cand+i);
         /* se incrementa en 1 el número de incógnitas agrupadas  */
         /* en cada subconjunto */
         *(C+(*(Procpe+(NP+1)*i+j)))=*(C+(*(Procpe+(NP+1)*i+j)))+1;
         /* se elimina a la incógnita como disponible             */
         *(Liber+(*(Cand+i)))=1;
         *(Peso+(*(Cand+i)))=0;
         /* se elimina a la incógnita como candidata              */
         *(Cand+i)=-1;
         }
      nagrup++; /* se incrementa en 1 el nro de incog. agrupadas  */
      i = NP + 1;
```

```
          }

        else         /* si no se realizara overlapping                  */
          {
          /* se asocia la incog al primer subconjunto que la selecciono */
          *(Magrup + (NN+1)*( *(Procpe+(NP+1)*i+1)) +
            *(C+(*(Procpe+(NP+1)*i+1)))+1)=*(Cand+i);


          /* se incrementa en 1 el número de incógnitas agrupadas       */
          /* en cada subconjunto                                        */
          *(C+(*(Procpe+(NP+1)*i+1)))=*(C+(*(Procpe+(NP+1)*i+1)))+1;
          /* se elimina a la incógnita como disponible                  */
          *(Liber+(*(Cand+i)))=1;
          *(Peso+(*(Cand+i)))=0;
          /* se elimina a la incógnita como candidata                   */
          *(Cand+i)=-1;
          nagrup++; /* se incrementa en 1 el nro de incog. agrupadas    */
          i = NP + 1;
          }
        }
      }
   }/* fin del while */

/* liberación de la memoria asignada dinámicamente                      */
delete(Procpe);
delete(Nprocpe);
delete(Macop);
delete(Cand);
delete(Nady);
delete(Pelea);
delete(Q);

}

/* ********************** fin de Etapa3 ****************************** */

/* ******************************************************************** */
/*   verigual                                                           */
/*   Función que verifica si todas las componentes de un vector son     */
/* iguales y retorna el mayor valor si son diferentes sus elementos     */
/* -------------------------------------------------------------------- */
/* Parámetros :   dim         : dimensión del vector                    */
/*                *vector     : vector al que se aplicará la función    */
/* -------------------------------------------------------------------- */
float verigual( int dim, float *vector )
{
int i,j, dim2;
float *auxv, max1;
auxv = new float[dim+1];

i=1;
j=1;
max1 = 0;

/* cargar el vector auxiliar con los elementos distintos de -1          */
for(i=1;i<=dim;i++)
   {
   if(*(vector+i)==-1)
     continue;          /* ignorar las componentes de valor -1          */
```

```
   else
      {
      *(auxv+j)=*(vector+i);
      j++;
      }
   }

dim2 = j-1;

/* buscar el mayor elemento del vector auxiliar                          */
max1 = *(auxv+1);
for(i=2;i<=dim2;i++)
   if( max1 < *(auxv+i))
      max1 = *(auxv+i);

/* ver si son iguales los elementos del vector aux                       */
for(i=2;i<=dim2;i++)
   if(*(auxv+1)!=*(auxv+i))
      return(max1);    /* no son iguales */

return (-1);            /* todos los cupos parciales son iguales       */

}
/* ********************* fin de verigual ***************************** */


/* ******************************************************************** */
/*   maxacop                                                            */
/*   Función que retorna el valor del mayor acoplamiento en un          */
/* subconjunto con otra incógnita dada                                  */
/* -------------------------------------------------------------------- */
/* Parámetros :   *mat  : matriz de agrupamientos                       */
/*                p     : índice que indica el subconjunto a analizar   */
/*                ninc  : cantidad de incógnitas de dicho subconjunto   */
/*                incog : incógnita externa a ser analizada             */
/* -------------------------------------------------------------------- */
float maxacop(int *mat, int p, int ninc , int incog )
{
int i,pos;
float max,val1,val2;

max=0;
for(i=1;i<=ninc;i++) /* para cada uno de los elementos del subconjunto */
   {
   /* hallar el valor del mayor acoplamiento                            */
   if(*(Mcoef+(NN+1)*incog+(*(mat+(NN+1)*p+i)))>max)
      {
      /* se verifica por filas                                          */
      max=*(Mcoef+(NN+1)*incog+(*(mat+(NN+1)*p+i)));
      pos=i; /* se guarda que incógnita es                             */
      }
   /* se verifica por columnas                                          */
   if(*(Mcoef+(NN+1)*(*(mat+(NN+1)*p+i))+incog)>max)
      {
      max=*(Mcoef+(NN+1)*(*(mat+(NN+1)*p+i))+incog);
      pos=i;       /* se guarda que incógnita es                       */
      }
   }
val1=fabs(*(Mcoef+(NN+1)*incog+(*(mat+(NN+1)*p+pos))));
val2=fabs(*(Mcoef+(NN+1)*(*(mat+(NN+1)*p+pos))+incog));
```

```
if(incog==-1)
  return 0;
else
  {
  if(val1>val2) return(val1);
  else return(val2);
  }
}
/* ********************** fin de maxacop ***************************** */


/* ********************************************************************** */
/*   cantad                                                               */
/*   Función que retorna la cantidad de incógnitas adyacentes a un        */
/* subconjunto dado                                                       */
/* ---------------------------------------------------------------------- */
/* Parámetros :   *mat  : matriz de agrupamientos                         */
/*                p     : subconjunto a ser analizado                     */
/*                ninc  : cantidad de incógnitas en dicho subconjunto     */
/* ---------------------------------------------------------------------- */
int cantad( int *mat, int p, int ninc )
{
int *Disp,
    nady,        /* número de adyacencias                                 */
    i,j;

nady=0;     /* inicialización del número de adyacencias                   */
Disp=new int[NN+1];     /* vector auxiliar de disponibilidad              */

for(i=1;i<=NN;i++)      /* inicialización de Free a 1                     */
   {
   *(Disp+i)=1;
   if(*(Liber+i)==1)   /* si la incógnita no se encuentra disponible     */
       *(Disp+i)=0;
   }

for(i=1;i<=ninc;i++)    /* para cada incógnita del subconjunto            */
   for(j=1;j<=NN;j++)  /* para cada incógnita del sistema                */
   /* si no fue aun contada                                               */
   /* tiene acoplamiento por filas o por columnas                         */
   if((*(Disp+j)==1) && (*(mat+(NN+1)*p+i)!=j) &&
       ((*(Mcoef+(NN+1)*(*(mat+(NN+1)*p+i))+j)!=0) ||
       (*(Mcoef+(NN+1)*j+(*(mat+(NN+1)*p+i)))!=0)))
     {
     *(Disp+j)=0;
     nady++;
     }
delete(Disp);   /* liberación de la memoria                               */
return(nady);
}

/* ********************** fin de cantad ***************************** */
```

```
/* ************************************************************** */
/* Etapa 4                                                        */
/* Función que calcula el parámetro de selección de la partición  */
/* -------------------------------------------------------------- */
/* Parámetros : config      : archivo de la partición             */
/*              opseleccion : opción de tipo de parámetro de selección */
/* -------------------------------------------------------------- */
```

```
float Etapa4(char *config, int opselecc)
{
FILE *arch_cf,    /* archivo donde se encuentra la partición          */
     *mah;    /* archivo donde se guardara la matriz de comparación   */
int i,j,k,l,auxi;

int np,       /* número de procesadores                               */
     *C,          /* número de incógnitas en cada procesador          */
     *magrup;     /* matriz que contiene la partición                 */

float *aii,*aij,*h, par_A, spec;

char cade[10];

/* lectura del archivo de partición                                   */
arch_cf=fopen(config,"r");

fscanf(arch_cf,"%d",&np);     /* se lee el número de procesadores     */
fscanf(arch_cf,"%s",&cade);

/* asignación dinámica de memoria                                     */
magrup=new int[(NN+1)*(np+1)];
C=new int[np+1];
h=new float[(np+1)*(np+1)];
for(i=1;i<=np;i++)            /* para cada procesador                 */
    {
    fscanf(arch_cf,"%s",&cade);
    fscanf(arch_cf,"%d",&auxi);
    fscanf(arch_cf,"%s",&cade);
    fscanf(arch_cf,"%d",(C+i));
    fscanf(arch_cf,"%s",&cade);
    for(j=1;j<=*(C+i);j++)
        {
        fscanf(arch_cf,"%d",(magrup+(NN+1)*i+j));
        }
    }
fclose(arch_cf);

if(opselecc==1)   /* si el parámetro será el Radio espectral          */
   {
   /* calculo de la matriz de comparación H                           */
   /* ----------------------------------------------------------------- */

   for(i=1;i<=np;i++) /* para cada una de las filas-bloque de la matriz */
     {
     aii=new float[(C[i]+1)*(C[i]+1)];
     blockd(Mcoef,magrup,*(C+i),aii,i); /* organiza el bloque diagonal */
     inva(*(C+i),aii);   /* invierte la matriz diagonal */
```

```
      for(j=1;j<=np;j++)       /* para cada bloque                        */
        {
        if(i==j)               /* el elemento hii = 0                     */
          {
          *(h+(np+1)*i+j)=0;
          continue;
          }
      aij=new float[(*(C+i)+1)*(*(C+j)+1)];
      blocknd(Mcoef,magrup,*(C+i),*(C+j),aij,i,j);
      *(h+(np+1)*i+j)=mult_norm(aii,aij,*(C+i),*(C+j));
      delete(aij);
      }
    delete(aii);
    }

    /* apertura del archivo donde se guardara la matriz de comparación   */
    /* ----------------------------------------------------------------- */
    mah=fopen("Mat_h.dat","w");

    /* se escribe en el la matriz H */
    fprintf(mah,"Matriz de comparación H \n\n");
    for(i=1;i<=np;i++)
      for(j=1;j<=np;j++)
       fprintf(mah,"H[%d][%d]=%f\n",i,j,*(h+(np+1)*i+j));

    /* calculo del radio espectral para las matrices de comparación      */
    /* de dimensión 2*/
    /* ----------------------------------------------------------------- */
    spec = pow((double)*(h+(np+1)*1+2) *
           (double)(*(h+(np+1)*2+1)),(double)0.5);
    fprintf(mah,"Radio espectral=%f\n",spec);
    return spec;

    fclose(mah);
    }

  if(opselecc==2)          /* si el parámetro será el Par_A               */
    {
    par_A=0;               /* inicialización del parámetro a cero         */
    for(i=1;i<=np;i++)     /* para cada subproblema                       */
      {
      /* para cada uno de los elementos del subproblema                   */
      for(j=1;j<=*(C+i);j++)
        {
        /* para cada uno de los demás subproblemas                        */
        for(k=1;k<=np;k++)
          {
          if(k==i)         /* no se considera a si mismo como subproblema */
            continue;
          /* para cada uno de los elementos del subproblema               */
          for(l=1;l<=*(C+k);l++)
            {
            par_A=par_A+fabs(*(Mcoef+ (NN+1)*(*(magrup+(NN+1)*i+j)) +
              *(magrup+(NN+1)*k+l) ) );
            }
          }
        }
      }
    return par_A;
    }
```

```
return 0;
}
/* ********************** fin de Etapa4 ***************************** */


/* ******************************************************************* */
/*   blockd                                                            */
/*   Función que organiza el bloque diagonal Aii                       */
/* ------------------------------------------------------------------- */
/* Parámetros :   matcoef : matriz M                                   */
/*                matgrup : matriz que contiene a la partición         */
/*                CI      : vector del número de incógnitas de cada    */
/*                          procesador                                 */
/*                p       : número de procesadores                     */
/* ------------------------------------------------------------------- */

void blockd(float *matcoef, int *matgrup, int CI, float *ad, int p)
{
int i,j;

for(i=1;i<=CI;i++)
   for(j=1;j<=CI;j++)
      {
      *(ad+(CI+1)*i+j)=*(matcoef+(NN+1)*(*(matgrup+(NN+1)*p+i))+
                     (*(matgrup+(NN+1)*p+j)));
      }
}
/* ********************** fin de blockd ***************************** */


/* ******************************************************************* */
/*   blocknd                                                           */
/*   Función que organiza el bloque no diagonal Aij                    */
/* ------------------------------------------------------------------- */
/* Parámetros :   matcoef : matriz M                                   */
/*                matgrup : matriz que contiene a la partición         */
/*                CI      : número de incógnitas del procesador "i"    */
/*                CJ      : número de incógnitas del procesador "j"    */
/*                f       : índice de fila                             */
/*                c       : índice de columna                          */
/* ------------------------------------------------------------------- */

void blocknd(float *matcoef, int *matgrup, int CI, int CJ, float *ad,
            int f, int c)
{
int i,j;

for(i=1;i<=CI;i++)
   for(j=1;j<=CJ;j++)
      {
      if(*(matgrup+(NN+1)*f+i)==*(matgrup+(NN+1)*c+j))
        {
        *(ad+(CJ+1)*i+j)=0;
        continue;
        }
      *(ad+(CJ+1)*i+j)=*(matcoef+(NN+1)*(*(matgrup+(NN+1)*f+i))+
                     (*(matgrup+(NN+1)*c+j)));
      }
}
/* ********************** fin de blocknd **************************** */
```

```
/* ********************************************************************* */
/*   mult_norm                                                           */
/*   Función que multiplica dos matrices y calcula la norma infinita     */
/* de la matriz producto                                                 */
/* --------------------------------------------------------------------- */
/* Parámetros :   mii, mij : matrices a multiplicar                      */
/*                CI,CJ    : dimensiones                                 */
/* --------------------------------------------------------------------- */

float mult_norm( float *mii, float *mij, int CI, int CJ )
{
int i,j,k;    /* variables de ciclo                                      */
float *sum,   /* vector de sumatorias por filas                          */
      max;    /* variable aux. para el calculo de la norma infinita      */

/* asignación dinámica de memoria para el vector sum                     */
sum=new float[CI];
/* inicialización de sum a cero                                          */
for(i=1;i<=CI;i++)
    *(sum+i)=0;

for(i=1;i<=CI;i++)      /* para cada fila de mii                         */
   for(j=1;j<=CJ;j++)   /* para cada columna de mij                      */
     for(k=1;k<=CI;k++)
          {
          *(sum+i)=*(sum+i)+fabs(*(mii+(CI+1)*i+k)*(*(mij+(CJ+1)*k+j)));
          }

/* calculo de la norma infinita del vector sum                           */
max=0;
for(i=1;i<=CI;i++)
   {
   if( fabs(*(sum+i)) > max)
     max=fabs(*(sum+i));
   }
delete(sum);
return max;
}

/* ********************** fin de mult_norm ************************** */
```

APÉNDICE B:
PROBLEMAS EJEMPLOS Y SU RESOLUCIÓN

PROBLEMAS EJEMPLOS Y SU RESOLUCIÓN

Como se ha visto en el capítulo 7, se resolvió el problema del Flujo de Potencia Eléctrica para problemas paradigmas de la IEEE (*The Institute of Electrical and Electronical Engineers*). En este apéndice se presenta una breve explicación del problema del Flujo de Potencia Eléctrica, así como una descripción de la forma en que éste fue resuelto para los sistemas eléctricos en estudio.

B.1 La red eléctrica: modelo matemático

La formulación de un modelo matemático adecuado a las características de la red y al tipo de estudio a ser realizado es el paso inicial para el análisis y resolución de un sistema eléctrico. El modelo matemático que aquí se presenta formulará las ecuaciones necesarias para proceder a la partición del sistema, asignando a los distintos procesadores del sistema distribuido las correspondientes ecuaciones a ser resueltas.

En la abundante bibliografía disponible sobre este punto particular [27] , se destacan dos formas de representar una red eléctrica: *ecuaciones de nudos* y *ecuaciones de lazos o mallas*. En lo que sigue de la sección se utilizará la primera de ellas.

Para este caso, las variables del sistema son las tensiones complejas en los nudos (módulo y fase) y las corrientes nodales.

De este modo, el conjunto completo de ecuaciones de nudos que definen una red se puede expresar en forma matricial como:

$$\begin{bmatrix} Y_{11} & Y_{12} & \cdots & Y_{1n} \\ Y_{21} & Y_{22} & \cdots & Y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ Y_{n1} & Y_{n2} & \cdots & Y_{nn} \end{bmatrix} \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ En \end{bmatrix} = \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{bmatrix} \qquad \text{(B.1)}$$

donde

$Y_{ii} = \sum_{\substack{m=0 \\ m \neq i}}^{n} y_{im}$ es la admitancia propia de la barra *i*.

$Y_{im} = y_{im}$ es la admitancia mutua entre las barras *i* y *m*.

La ecuación matricial (B.1.) puede ser expresada en la forma:

$$\mathbf{YE} = \mathbf{I}(\mathbf{E}) \qquad \text{(B.2)}$$

donde

$\mathbf{Y} \subset C^{n \times n}$ es la matriz admitancia

$\mathbf{I} \subset C^{n}$ es el vector de corrientes inyectadas

$\mathbf{E} \subset C^{n}$ es el vector de tensiones (módulo y fase)

La matriz **Y** es compleja, extremadamente esparza, simétrica, no posee estructura definida y sus elementos proporcionan información sobre las ligaciones entre los nudos del sistema eléctrico.

B.2 Flujo de Potencia en un sistema eléctrico

Para poder definir con propiedad el problema del Flujo de Potencia, se deben identificar cuatro variables en cada barra *i* del sistema:

$\mathbf{P}_i$ = Potencia real o activa,

$\mathbf{Q}_i$ = Potencia reactiva o de cuadratura,

$\mathbf{V}_i$ = Módulo de la tensión E_i,

θ^i = Fase de la tensión E_i.

Inicialmente sólo se conocen dos de las cuatro variables para cada barra *i*, y el **objetivo** de la resolución del Flujo de Potencia es calcular las otras dos variables implicadas en el problema.

Utilizando una variante del método de Newton-Raphson, el algoritmo (2.35) puede ser escrito en su forma matricial de la siguiente manera:

$$\begin{bmatrix} \nabla \mathbf{P}(k) \\ \nabla \mathbf{Q}(k) \end{bmatrix} = \begin{bmatrix} \mathbf{H}(k) & \mathbf{N}(k) \\ \mathbf{J}(k) & \mathbf{L}(k) \end{bmatrix} \begin{bmatrix} \nabla \theta(k) \\ \dfrac{\nabla \mathbf{V}(k)}{\mathbf{V}(k)} \end{bmatrix} \qquad \text{(B.3)}$$

donde:

$\nabla \mathbf{P}$ = vector de errores de P

$\nabla \mathbf{Q}$ = vector de errores de Q

$\nabla \theta$ = vector de correcciones de θ

$\nabla \mathbf{V}$ = vector de correcciones de **V**

y las submatrices **H**, **N**, **J** y **L** son las componentes de la matriz jacobiana.

En la sección 2.4 se han expuesto con detalle los pasos a seguir para aplicar el método de Newton-Raphson. A cada iteración, los valores de los módulos y las fases de

las tensiones se actualizan al sumar los incrementos calculados a partir de (B.3) y se verifica si el método ya llegó a la solución comparando las potencias calculadas con las potencias que son datos del sistema.

Para una mejor comprensión, en la figura B.1 se esquematiza el programa de resolución de Flujo de Potencia Eléctrica.

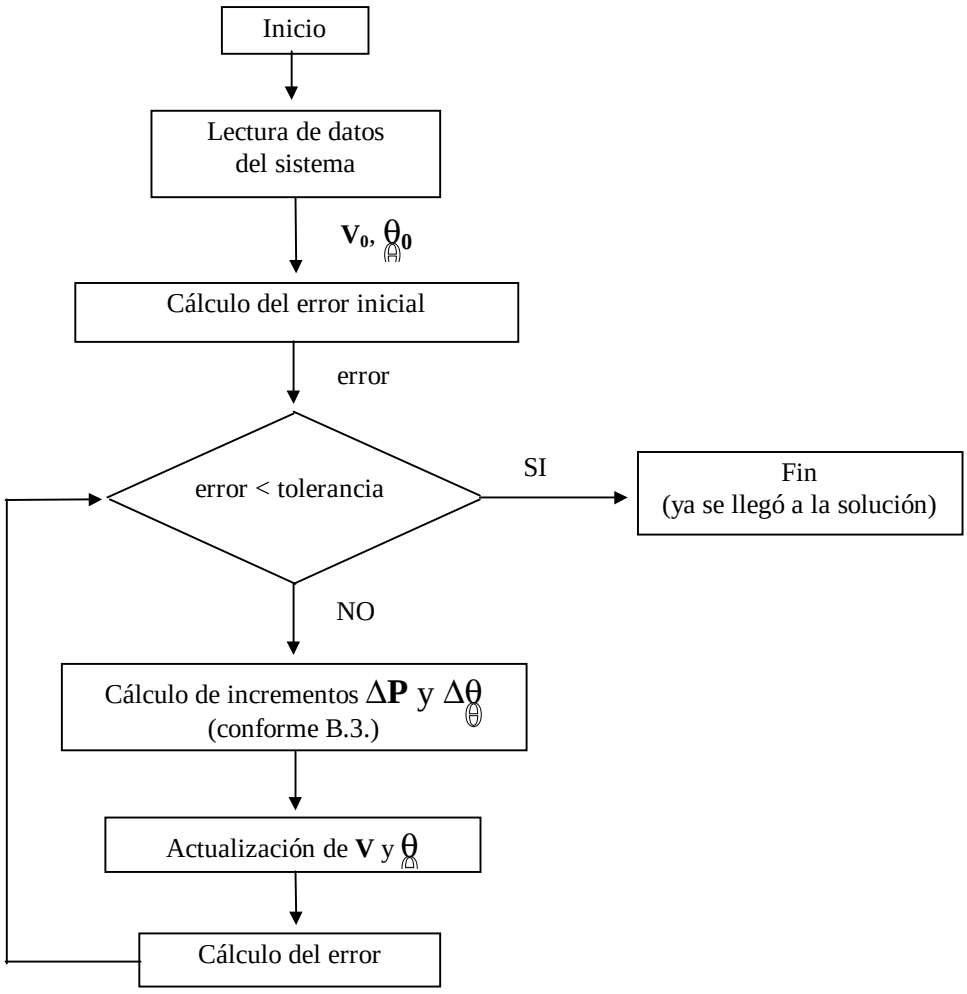


Figura B.1 : Diagrama de flujo del programa de resolución de Flujo de Potencia Eléctrica.

B.3 Resolución distribuida del Flujo de Potencia

En el contexto del procesamiento paralelo, donde varios procesadores resuelven su subproblema local, es necesario asignar a cada procesador una subred eléctrica determinada. Una vez que esto fue realizado, cada procesador resolverá de forma local el problema del Flujo de Potencia para su subred (subsistema), asumiendo conocidas las variables de las demás barras del sistema eléctrico.

En concordancia con lo expuesto en la formulación matemática del Capítulo 2, el sistema global de ecuaciones a ser resuelto tendrá la forma

$$\Phi(\mathbf{x}) = \mathbf{S} - \mathbf{E}\mathbf{I}^{*} = \mathbf{0} \tag{B.4}$$

donde

$$\mathbf{x} = \begin{bmatrix} \delta_1 \\ \vdots \\ \delta_n \\ V_1 \\ \vdots \\ V_n \end{bmatrix}, \quad \delta_i, V_i \in \Re; \quad \mathbf{S} = \begin{bmatrix} S_1 \\ \vdots \\ S_n \end{bmatrix}, \quad S_i = P_i + j\,Q_i \in C, \tag{B.5}$$

$$\mathbf{E}\mathbf{I}^{*} = \begin{bmatrix} E_1 \sum_{m \in 1} Y_{1m}^{*} E_m^{*} \\ \vdots \\ E_n \sum_{m \in n} Y_{nm}^{*} E_m^{*} \end{bmatrix}, \qquad E_i, Y_{im} \in C \tag{B.6}$$

y E_m^{*} es el complejo conjugado de E_m.

Una vez que un subsistema haya sido asignado a cada procesador *i*, el problema local a ser resuelto por dicho procesador *i* será, conforme (2.11) y (2.12):

$$\Phi_i(\mathbf{x}) = \mathbf{S}_i - \mathbf{E}_i \mathbf{I}_i^* = \mathbf{0} \qquad \text{(B.7)}$$

con $\mathbf{S}_i$, $\mathbf{E}_i$, $\mathbf{I}_i^* \in C^{n_i}$, donde n_i es el número de barras (incógnitas) asignadas al procesador *i*.

Tenemos así que cada procesador *i* actualiza las variables correspondientes al subsistema eléctrico a él asignado, utilizando el algoritmo (2.38), para lo cual se necesitan los valores de tensión calculados por los otros procesadores, pues estos son necesarios en el cálculo de $\Delta\mathbf{P}$ y $\Delta\mathbf{Q}$.

Entonces, en cada iteración, y a partir de los incrementos calculados, cada procesador *i* actualiza sus incógnitas locales θ_i y $\mathbf{V}_i$, comunicando sus resultados parciales a los demás procesadores del sistema distribuido, avanzando así hacia la solución global del problema.

En la figura B.2. podemos observar como cada procesador utiliza la fórmula (B.7) para actualizar su incógnita local x_i utilizando los resultados actualizados por los otros procesadores del sistema, que les fueron comunicados a través del sistema de comunicación.

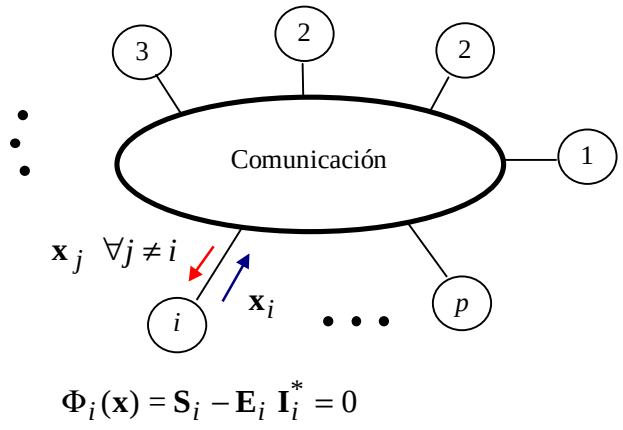


Figura B.2 : Método de Newton-Raphson implementado en un sistema de *p* procesadores para la resolución del problema de Flujo de Potencia Eléctrica.

BIBLIOGRAFÍA

[1] Aboytes, F. y Sasson, A. M., "A power Systems Decomposition Algorithm", *Proceedings of the IEEE Power Industries computer Application Conference*, pp. 448-452, 1971.

[2] Barán B., Cardozo F., Atlasovich J. y Schaerer C., "Solving the point of Collapse Problem using a Heterogeneus Computer Network". International Conference on Information Systems Analysis and Synthesis. Orlando, EE.UU, julio 1996.

[3] Barán B., *Estudio de Algoritmos Combinados Paralelos Asíncronos*. Tesis Doctoral COPPE/UFRJ. Río de Janeiro, Brasil, octubre 1993.

[4] Barán B., Kaszkurewicz E. y Bhaya A., "Parallel Asynchronous Team Algorithms: Convergence and Performance Analysis". *IEEE Transactions on Parallel & Distributed Systems*, julio 1996.

[5] Barán B., Kaszkurewicz E. y Falcão D.M., "Team Algorithm in Distributed Load Flow Computations", *IEE Proceeding on Generation, Transmission and Distribution*, vol. 142, no. 6, pp. 583-588, noviembre 1995.

[6] Bertsekas D.P. y Tsitsiklis J.N. *Parallel and Distributed Computation. Numerical Methods.* Editorial Prentice-Hall. 1989.

[7] Bhaya M., Kaszkurewicz E. y Mota F., "Asynchronous Block-Iterative Methods for Almost Linear Equations". *Linear Algebra and Its Applications*, vol. 155, pp. 487-508, 1991.

[8] *Cabling Business Magazine*, vol. 6 no. 6, junio 1996.

[9] Carré B.A., "Solution of Load-Flow by Partitioning Systems into Trees", *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-88, pp. 1931-1938, noviembre 1968.

[10] Ikeda M. y Šiljak D.D., Overlapping decomposition, expansions and contractions of dynamic systems. *Large Scale System 1*, North-Holland Publishing Co., pp.29-38, 1980.

[11] INTEL Corporation, "Intel Microprocessor Quick Reference Guide". Documento disponible en World Wide Web: http :// www. intel. com /pressroom /no_frame/ quickref.htm

[12] Irving M.R. y Sterling M.J.H., "Optical Network Tearing Using Simulated Annealing", *IEEE Proceedings,* vol. 137, no. 1, pp. 69-72, enero 1990.

[13] Jain M.K. y Rao N.D., "A Power System Networks Decomposition for Network Solutions", *IEEE Transactions on Power Apparatus and Systems,* vol. PAS-92, no. 2, pp. 619-625, 1973.

[14] Kaskurewicz E., Bhaya A. y Šiljak D. D. "On the convergence of parallel asynchronous block-iterative computations", *Linear Algebra Appl.*, 131, pp. 139-160, 1990.

[15] Mickle M.H., Vogt W.G. y Colclaser R.G., "Paralel Processing and Optimal Network Decomposition Applied to Load Flow Analysis and Related Problems", *Special Report of the Electrical Power Research Institute,* EPRI EL-566-SR, pp. 171-182, 1977.

[16] Monticelli A,. *Fluxo de carga em redes de energía elétrica*. Editora Edgard Blucher Ltda, 1983.

[17] Ogbuobiri E., Tinney W.F., y Walker J.W., "Sparsity Oriented Decomposition for Gaussian Elimination on Matrices", *IEEE Transactions on power Apparatus ans Systems,* vol. PAS-89, no. 1, pp. 141-150, enero 1970.

[18] Press W.H., Flanney B.P., Teokolsky S.A. y Vetterling W.T., *Numerical Recipes in C - The Art of Scientific Computing*, Cambridge University Press, 1988.

[19] PVM: Parallel Virtual Machine. Documento disponible en World Wide Web: http: //www.epm.ornl.gov/pvm/

[20] Saheh A.O.M. y Laughton M.A.,"Cluster Analysis of Power System Networks for Array Processing Solutions", *IEEE Proceedings,* vol. 132, no. 4, pp. 172-178, julio 1985.

[21] Sangiovanni-Vincentelli A., Chen L.K., y Chua L.O., "An Efficient Heuristic Cluster algorthm for Tearing Large-Scale Networks", *IEEE Transactions on Circuits and Systems,* vol. CAS-89, no. 12, pp. 709-717, diciembre 1977.

[22] Sasson A.M., "Decomposition Tecnique Applied to the Nonlinear Programming Load-Flow Method", *IEEE Transactions on Power Apparatus and Systems,* vol. PAS-89, no. 1, pp. 78-82, enero 1970.

[23] Schaerer C. y Atlasovich J., *Flujo de Potencia Eléctrica en torno al Punto Crítico*, Tesis de grado, Facultad de Ingeniería de la Universidad Nacional de Asunción, mayo 1995.

[24] Sezer M. y Šiljak D.D., "Nested epsilon decompositions and clustering of complex systems", *Automática*, vol. 22, no. 3, pp. 69-72, 1986.

[25] Sezer M. y Šiljak D.D., "Nested epsilon decompositions of complex systems". IFAC 9th *World Congress*, Budapest, Hungría, julio 1984.

[26] Sezer M. y Šiljak D.D., "Nested epsilon decompositions of linear systems: Weakly coupled and overlapping blocks", *SIAM Journal of Matrix Analysis and Aplications*, 12, pp. 521-533, 1991.

[27] Stott B., "Review of load-flow calculation methods", *Proceedings of the IEEE*, 62, pp. 916-929, 1974.

[28] Undrill J.M. y Happ H.H., "Automatic Sectionalization of Power System Networks for Network Solution", *IEEE Transactions on Power Aparatus and Systems,* vol. PAS-90, no.1, pp. 43-53, enero / febrero 1971.

[29] Vale M.H., Falcão D.M. y Kaszkurewicz E., "Electrical Power Network Decomposition for Parallel Computations". *IEEE Iternational Symposium on Circuits and Systems*-ISCAS 92. San Diego, California, 1992.

[30] Vale, M.H. *Descomposicao de Redes Elétricas para Processamento Paralelo.* Tesis Doctoral COPPE/UFRJ. Río de Janeiro, Brasil, 1994.

[31] Yourdon E. *Analise Estructurada Moderna.* Editora Campus, 1990.

[32] Zecevic A.Y. y Siljak D.D., "Balanced Decompositions of Sparse Systems for Parallel Processing", *IEEE Transactions on Circuits and Systems,* vol. 41, no. 3, pp. 220-233, marzo 1994.