Partición de Sistemas Lineales

Para su Resolución en Sistemas Distribuidos

Benjamín Barán

Centro Nacional de Computación.

Universidad Nacional de Asunción.

Fax: (595) (21) 585551 E-mail: bbaran@una.py

y

Facultad de Ciencias y Tecnología

Universidad Católica Nuestra Señora de la Asunción

Diana Benítez

Facultad de Ciencias y Tecnología
Universidad Católica Nuestra
Señora de la Asunción

Rodrigo Ramos

Centro Nacional de Computación
Universidad Nacional de Asunción

Este trabajo fue realizado en el marco del proyecto *Computación Distribuida* patrocinado por UNESCO, contrato ROSTLAC 885.610-4; con el auspicio de la Universidad Nacional de Asunción (UNA) y la Universidad Católica Nuestra Señora de la Asunción (UCA).

OBJETIVO PRINCIPAL

El objetivo del presente trabajo es descomponer un sistema de ecuaciones lineales en subsistemas menores de forma a minimizar el número de iteraciones y el tiempo empleado en su resolución utilizando un sistema distribuido heterogéneo.

El problema que se plantea consiste en descomponer un sistema de n ecuaciones con n incógnitas en p procesadores ($p \leq n$), de forma tal que las cargas computacionales sean proporcionales a la performance de cada procesador y que la dependencia entre las variables actualizadas por los diferentes procesadores no dificulte la convergencia.

BREVE DESCRIPCIÓN

La presente obra plantea una propuesta para descomponer sistemas de ecuaciones lineales de forma tal que puedan ser resueltos en ambientes distribuidos heterogéneos. La obra está basada en una metodología de descomposición propuesta por Vale M. et al. en 1992. Dicha metodología permite dividir un sistema lineal de ecuaciones en subsistemas menores de aproximadamente igual dimensión, para su resolución en sistemas computacionales conformados por procesadores similares (de igual performance).

La presente propuesta consiste en variar la metodología anterior de descomposición, asignando a los procesadores del sistema distribuido distintos números de ecuaciones, según la perfomance relativa de cada uno de los procesadores. Es decir, se pueden obtener particiones para sistemas computacionales homogéneos o heterogéneos, según sea la necesidad del usuario.

Se presentan resultados experimentales que avalan la validez la nueva propuesta.

1. INTRODUCCIÓN

Las implementaciones paralelas de algoritmos iterativos se están volviendo una opción importante dentro del contexto de la computación distribuida gracias a las diversas características que éstas poseen, entre las cuales pueden citarse la facilidad de implementación, balanceamiento de carga computacional según la performance relativa de los procesadores y los menores tiempos de ejecución.

Para aprovechar estas características, es necesario dividir un problema en subproblemas, los cuales son asignados para su resolución en forma paralela a los distintos procesadores del sistema distribuido. De esta forma se puede operar inclusive en un ambiente computacional heterogéneo (procesadores de diversas performance).

Una vez que se dispone de un sistema distribuido con procesadores conocidos, es necesario descomponer el problema en distintos subproblemas de forma que la implementación del algoritmo de resolución sea computacionalmente eficiente.

La descomposición está condicionada por dos factores:

a) las capacidades de procesamiento relativo de cada máquina del sistema distribuido. Este factor determina las dimensiones de los subproblemas asignados a cada procesador;
b) el grado de acoplamiento que puedan tener los subproblemas entre sí. Esto último determina la dependencia entre las variables de los subproblemas y por consiguiente, influye en la velocidad de convergencia del algoritmo [1].

El problema que se plantea consiste en descomponer un sistema de n ecuaciones con n incógnitas en p procesadores, donde $p \leq n$, de forma tal que a cada procesador se le asigne un subproblema proporcional a su performance, y que la dependencia entre las variables actualizadas por cada uno de los procesadores no dificulte la convergencia.

Existen diversos métodos de partición ya publicados. Sin embargo, los trabajos existentes presentan características apropiadas a diferentes aplicaciones específicas. Varios trabajos buscan la eficiencia de técnicas de resolución bien definidas como, por ejemplo, la técnica diacóptica [17,6], la eliminación de Gauss [3] y la programación no lineal [12]. Por lo tanto, éstos métodos no están directamente relacionados a la aplicación de procesamiento paralelo, objetivo de este trabajo.

Entre los trabajos dedicados a las aplicaciones del procesamiento paralelo hay varias propuestas interesantes, tales como [8,5,11,19]. Dichos trabajos, sin embargo, no incluyen con claridad un análisis de los aspectos básicos que favorecen el desempeño de los métodos de resolución que irán a utilizar el sistema descompuesto. Una primera metodología de descomposición que considera este análisis es la conocida Descomposición ε [13-16].

La Descomposición ε es la metodología más utilizada comúnmente. Esta metodología se basa en la construcción de un grafo a partir de la matriz de coeficientes del sistema; las incógnitas del sistema son representadas por los nodos del grafo y los coeficientes del sistema son las ramas del grafo. Nótese que los elementos de la diagonal principal de la matriz de coeficientes del sistema no se incluyen en el grafo, ya que éstos no representan relaciones entre incógnitas distintas. Una vez que se cuenta con el grafo del sistema, se eliminan las ramas que poseen un valor menor que un límite inferior predeterminado, y de esta forma se obtienen las particiones deseadas. En algunos casos, el límite inferior tiene que ser variado empíricamente hasta obtener la partición deseada.

Sin embargo, este método no presenta ciertas características apropiadas a la utilización de procesamiento paralelo, tales como el control sobre el número de subproblemas en el cual el sistema es descompuesto y el control sobre el tamaño de los mismos; es decir, no siempre produce las particiones deseadas conforme a la performance entre los procesadores.

Surge entonces la necesidad de un método computacional más eficiente que la tradicional Descomposición ε. Una propuesta en este sentido es la presentada a continuación.

En la sección 2, se describe el método iterativo en el proceso resolutivo. En la sección 3, se presentan el principio básico del método y una explicación detallada del mismo. Resultados experimentales se presentan en la sección 4 y las conclusiones en la sección 5.

2. FORMALIZACIÓN DEL PROBLEMA

Considerando un sistema de n ecuaciones lineales con n incógnitas representadas por el vector $\boldsymbol{x}$, se tiene:

$$\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}, \qquad \boldsymbol{A} \in \mathrm{R}^{nxn}, \qquad \boldsymbol{x}, \boldsymbol{b} \in \mathrm{R}^{n} \tag{1}$$

donde $\boldsymbol{A}$ y $\boldsymbol{b}$ son datos del problema, siendo $a_{ij} = a_{ji}$ en el marco del presente trabajo. Note que las matrices simétricas se aplican a la resolución de importantes problemas de ingeniería como, por ejemplo, sistemas eléctricos.

La idea básica para resolver (1) en un sistema distribuido es usar p procesadores donde $p \leq n$, de manera tal que cada procesador resuelve sólo una parte del sistema de ecuaciones y comunica sus resultados parciales a los demás procesadores, resolviendo en forma conjunta el problema global.

Para esto, la matriz A puede ser descompuesta de la siguiente manera:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{11} \cdots \boldsymbol{A}_{1p} \\ \vdots \quad \ddots \quad \vdots \\ \boldsymbol{A}_{p1} \cdots \boldsymbol{A}_{pp} \end{bmatrix}$$

(2)

siendo $\boldsymbol{A} = (\boldsymbol{A}_{ij}) \in \mathsf{R}^{n \times n}$, donde $\boldsymbol{A}_{ij} \in \mathsf{R}^{n_i \times n_j}$ y $i, j \in \{1, \ldots, p\}$

Haciendo $\boldsymbol{A} = \boldsymbol{S} - \boldsymbol{T}$ donde

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{A}_{11} & 0 & \cdots & 0 \\ 0 & \boldsymbol{A}_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{A}_{pp} \end{bmatrix}$$

(3)

y

$$\boldsymbol{T} = -\begin{bmatrix} 0 & \boldsymbol{A}_{12} & \cdots & \boldsymbol{A}_{1p} \\ \boldsymbol{A}_{21} & 0 & \cdots & \boldsymbol{A}_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}_{p1} & \boldsymbol{A}_{p2} & \cdots & 0 \end{bmatrix}$$

(4)

se obtiene el siguiente método iterativo:

$$\boldsymbol{x}(k+1) = \boldsymbol{S}^{-1}\, \boldsymbol{T}\, \boldsymbol{x}(k) + \boldsymbol{S}^{-1}\, \boldsymbol{b}$$

(5)

$$\begin{bmatrix} \boldsymbol{x}_1(k+1) \\ \boldsymbol{x}_2(k+1) \\ \vdots \\ \boldsymbol{x}_p(k+1) \end{bmatrix} = \begin{bmatrix} -\boldsymbol{A}_{11}^{-1} (\boldsymbol{A}_{12}\, \boldsymbol{x}_2(k) + \cdots + \boldsymbol{A}_{1p}\, \boldsymbol{x}_p(k)) \\ -\boldsymbol{A}_{22}^{-1} (\boldsymbol{A}_{21}\, \boldsymbol{x}_1(k) + \cdots + \boldsymbol{A}_{2p}\, \boldsymbol{x}_p(k)) \\ \vdots \\ -\boldsymbol{A}_{pp}^{-1} (\boldsymbol{A}_{p1}\, \boldsymbol{x}_1(k) + \cdots + \boldsymbol{A}_{p(p-1)}\, \boldsymbol{x}_{p-1}(k)) \end{bmatrix} + \begin{bmatrix} \boldsymbol{A}_{11}^{-1}\, \boldsymbol{b}_1 \\ \boldsymbol{A}_{22}^{-1}\, \boldsymbol{b}_2 \\ \vdots \\ \boldsymbol{A}_{pp}^{-1}\, \boldsymbol{b}_1 \end{bmatrix}$$

(6)

De las ecuaciones (5) y (6) podemos escribir el algoritmo iterativo de la siguiente forma:

$$\boldsymbol{x}_i(k+1) = -\sum_{\substack{j=1 \\ j\neq i}}^{j=p} \boldsymbol{A}_{ii}^{-1}\boldsymbol{A}_{ij}\,\boldsymbol{x}_j(k) + \boldsymbol{A}_{ii}^{-1}\,\boldsymbol{b}$$

(7)

El algoritmo (7) converge a la solución si se cumple la condición de bloco-diagonal dominancia [2], es decir:

$$\left(\left\|\boldsymbol{A}_{ii}^{-1}\right\|\right)^{-1} \geq \sum_{j=1;\, j\neq i}^{p} \left\|\boldsymbol{A}_{ij}\right\| \qquad \forall\ i,j \in \{1,\ldots,p\}$$

(8)

conforme definición dada en [4].

Se define además $\boldsymbol{w} \in \mathrm{N}^{\mathrm{p}}$ 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 si $\boldsymbol{w} = \begin{bmatrix} 1 & 2 & 4 \end{bmatrix}^T$ esto implica que el procesador 2 puede procesar el doble de ecuaciones que el procesador 1 en el mismo periodo de tiempo.

En este trabajo se pretende, dados el sistema de ecuaciones y un sistema distribuido con performance relativa $\boldsymbol{w}$, hallar una partición que resuelva el problema en forma eficiente; esto es, en un número de iteraciones razonablemente cercano al óptimo.

Para esto debe determinarse un vector $\boldsymbol{n} = \begin{bmatrix} n_1 & \cdots & n_p \end{bmatrix}^T$, donde los $n_i \in \mathrm{N}$ indican cuantas ecuaciones serán asignadas al procesador *i*, de manera que el valor de sus componentes sea proporcional al desempeño de cada procesador. Es decir, $\boldsymbol{n}$ y $\boldsymbol{w}$ deben acercarse lo más posible a la siguiente relación:

$$\boldsymbol{n} = \alpha\, \boldsymbol{w}$$

(9)

donde $\alpha \geq 1$ es una constante.

Además, debe verificarse que el valor de los $\| A_{ij} \|$, $\forall\ i \neq j$, sea lo más pequeño posible, de forma a facilitar la convergencia.

3. PROPUESTA DEL TRABAJO

3.1. Principio básico de la metodología

El método de descomposición propuesto utiliza el valor de los coeficientes del sistema de ecuaciones (Matriz ***A***) para establecer una medida del grado de acoplamiento entre las incógnitas. Se buscó una metodología para separar el sistema en los puntos donde dicho acoplamiento sea lo más débil posible.

El principio básico del método es la formación de *p* subconjuntos a partir de *p* incógnitas iniciales, llamadas *semillas*, donde *p* es el número pre-especificado de procesadores que se tiene en el sistema distribuido.

La elección de las semillas se realiza en base al *"peso"* de las incógnitas.

Sean $M = \frac{1}{nc} \sum_{i=1}^{n} \sum_{j=1,\, j\neq i}^{n} a_{ij} \qquad \forall a_{ij} \neq 0$ y $z_{ij} = a_{ij} / M$

donde M es media aritmética total, las a_{ij} son los elementos de la matriz A y $nc = nro\ de\ a_{ij} \neq 0$

El peso *P* de una incógnita *i* se calcula según la ecuación:

$$P_i = \sum_{j=1\ j\neq i}^{j=n} a_{ij}^{\,z_{ij}} \qquad \forall i \in \{1,\ldots,n\}, \qquad \forall a_{ij} \neq 0$$

(9)

Se introduce aquí la condición de *adyacencia* de dos incógnitas: las incógnitas ***i*** y ***j*** son adyacentes si $a_{ij} \neq 0$.

Una vez que se tienen determinadas las semillas de cada subsistema de acuerdo al algoritmo de selección de semillas, el cual será presentado en la sección 3.2 (Algoritmo 2), es necesario adicionar las demás incógnitas al subsistema adecuado. La forma de realizar esto es adhiriendo la variable más pesada adyacente a cada subproblema en formación.

El proceso de agrupamiento de incógnitas termina cuando todas las incógnitas han sido asignadas a algún subproblema

3.2. Descripción de la metodología

Basada en el principio descripto anteriormente, se propone una metodología compuesta de cuatro etapas que puede ser esquematizada conforme a la figura 1.

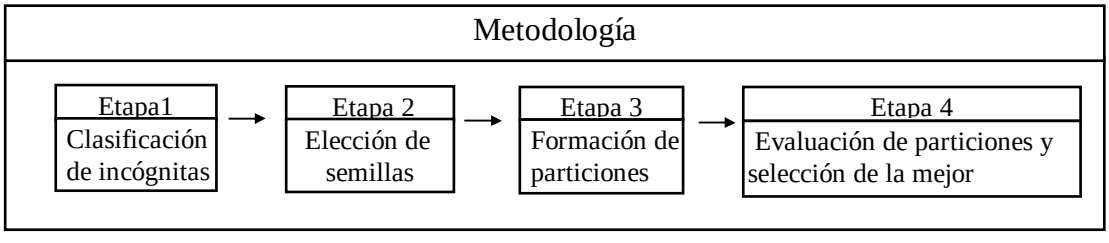


Figura 1: Etapas de la Metodología propuesta.

Etapa 1: Clasificación de incógnitas.

La clasificación de las incógnitas es hecha calculando el peso de cada incógnita y estableciendo un ranking de las mismas conforme sus pesos P_i. Es decir:

```
Desde i = 1 hasta n                    /* para cada una de las incógnitas */
  Calcular Pi según la ecuación (9)
  Hacer un ranking de los pesos Pi
```

Algoritmo 1

Etapa 2: Elección de semillas.

Antes de expresar el algoritmo de selección de semillas, es necesario definir los siguientes parámetros: *vlim*, *ngrup* y *nviz*.

Determinado un valor límite de peso *vlim* , las incógnitas candidatas a semilla son aquellas que posean pesos superiores a él. Teóricamente, todas las incógnitas podrían ser evaluadas como candidatas a semillas haciendo *vlim* = 0, pero esto implicaría un alto costo computacional. Esto es 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 candidatos a semillas a una cantidad razonable[18].

Se define como *ngrup* al número de incógnitas a ser agrupadas alrededor de cada candidata a semilla, en el Algoritmo 2. *Nviz* es un valor que se utiliza para evitar que incógnitas fuertemente acopladas entre sí sean semillas al mismo tiempo.

A continuación se presenta el algoritmo de selección de semillas:

```
Determinar las ternas {vlim, ngrup, nviz};
Para cada terna seleccionada
   Inicializar el conjunto K como vacío;      /* Conjunto de incógnitas candidatas a semillas
   */
   Inicializar el conjunto S como vacío;      /* Conjunto de semillas de una partición */
   Para cada una de las incógnitas i
      Si su peso Pi ≥ vlim  entonces
         Incluir la incógnita i  en K ;
   Para cada una de las incógnitas  i  en K
      Inicializar un conjunto I; como vacío;  /* Conjunto de incógnitas agrupadas alrededor
      de                                      las candidatas a semillas */
      Incluir la incógnita i  en I;
      Inicializar un conjunto CIA  como vacío; /* Conjunto de Incógnitas Adyacentes
                                                  al conjunto I*/
      Incluir en CIA  las incógnitas adyacentes a la incógnita i;
      Desde 1 hasta ngrup
         Incluir la incógnita de mayor peso del CIA en I;
         Eliminar esta incógnita del CIA;
         Incluir en CIA las incógnitas adyacentes a la recientemente incorporada en I,
```

/* que no se encuentren en ***CIA*** ni en ***I***; *******/*

Para cada conjunto ***I***

Calcular la sumatoria de los pesos de todas sus incógnitas ;

Seleccionar de ***K*** la incógnita que posea mayor sumatoria de peso y elegirla como primera semilla;

Incluir esta incógnita en ***S***;

Eliminar esta incógnita de ***K***;

Mientras el número de semillas ≤ *p*

Seleccionar en ***K*** la incógnita que posea la mayor sumatoria de pesos;

Si ésta no pertenece a las *nviz* primeras de los conjuntos ***I*** de las semillas ya seleccionadas

Elegir esta incógnita como semilla;

Incluir esta incógnita en ***S***;

Eliminar esta incógnita de ***K***;

De lo contrario

Eliminar esta incógnita de ***K***;

Algoritmo 2

Etapa 3: Formación de particiones.

La formación de particiones se realiza siguiendo los pasos del algoritmo 3.

Para cada grupo ***S***
  Definir los vectores ***C*** y $\boldsymbol{\varphi}$;
  /* $\mathbf{C} \in N^p$,$\mathbf{C} = [\ c_1, ..., c_p\ ]^T$ es un vector de estado donde los c_i representan la cantidad de incógnitas agrupadas en el conjunto I_i en un momento dado y $\boldsymbol{\varphi} \in N^p$, $\boldsymbol{\varphi} = [\varphi_1, ..., \varphi_p\ ]^T$ es otro vector de estado donde los φ_i se obtienen según $\varphi_i = \text{int}\ (c_i / w_i)$ */
  Para cada una de las semillas de ***S***
    Inicializar un conjunto ***I*** como vacío;
    Incluir en ***I*** la semilla *i*;
    Inicializar un conjunto ***CIA*** como vacío;
    Incluir en ***CIA*** las incógnitas adyacentes a la semilla i;
  Calcular los vectores ***C*** y $\boldsymbol{\varphi}$
  Mientras existan incógnitas no agrupadas
    Para todos los conjuntos $\boldsymbol{I}_i$ tales que $\varphi_i < \| \varphi \|_\infty$ ó $(\varphi_i = \varphi_1 = \varphi_2 = ... = \varphi_p)$;
      Seleccionar como incógnita candidata a ser incluida a la de mayor peso en el ***CIA*** correspondiente;
      Si existe coincidencia de incógnitas candidatas a ser incluidas
        Seleccionar el valor del mayor acoplamiento entre las incógnitas ya incluidas en $\boldsymbol{I}_i$ y la incógnita candidata a ser incluida;
        Incluir la incógnita candidata en el conjunto ***I*** con el cual posea mayor acoplamiento[1]
        Eliminar esta incógnita de todos los ***CIA***s;
      De lo contrario
        Incluir las incógnitas candidatas en los conjuntos ***I*** correspondientes;
        Eliminar estas incógnitas de todos los ***CIA***s;
    Incluir en los ***CIA***s correspondientes las incógnitas adyacentes a las incluidas;
    Actualizar ***C*** y $\boldsymbol{\varphi}$;

[1] Si existe nuevamente coincidencia, la asignación de la candidata se realiza en base a otros criterios.

Algoritmo 3

Etapa 4: Evaluación de particiones y selección de la mejor partición

Partiendo de distintas ternas de parámetros *vlim, ngrup* y *nviz*, pueden ser determinados varios conjuntos diferentes de semillas, que a su vez servirán de pie para generar diferentes particiones. Para identificar a la mejor partición, se impone aquí una selección de descomposiciones. Para ello se calcula para cada partición un cierto parámetro que se llamará en el contexto de esta obra "Parámetro de Acoplamiento A", el cual consiste en la sumatoria de todos los valores absolutos de las ligaciones entre incógnitas separadas por la partición. Es decir:

$A = \Sigma\ |a_{ij}|$ para todo par de incógnitas (x_i, x_j) separadas por la partición.

Será elegida como óptima aquella partición que presente el menor valor del parámetro A arriba definido.

3.3 Ejemplo

Particionar el siguiente sistema de ecuaciones de manera a resolverlo utilizando 2 procesadores con $\boldsymbol{w} = [4\ 1]^T$

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

donde

$$\boldsymbol{A} = \begin{bmatrix} 100 & 0.2 & 0 & 1 & 0.5 & 0 & 1.2 & 2 & 1.5 & 2 \\ 0.2 & 12 & 0.1 & 0 & 3 & 0 & 1.1 & 1 & 1.2 & 0 \\ 0 & 0.1 & 132 & 1 & 0.1 & 1 & 0 & 0 & 2 & 0.1 \\ 1 & 0 & 1 & 25 & 0 & 1 & 2 & 1.1 & 1 & 0.2 \\ 0.5 & 3 & 0.1 & 0 & 80 & 1 & 0.1 & 0 & 2 & 2 \\ 0 & 0 & 1 & 1 & 1 & 100 & 0.1 & 0.1 & 0.5 & 2 \\ 1.2 & 1.1 & 0 & 2 & 0.1 & 0.1 & 12 & 0 & 0.1 & 0 \\ 2 & 1 & 0 & 1.1 & 0 & 0.1 & 0 & 50 & 0 & 2 \\ 1.5 & 1.2 & 2 & 1 & 2 & 0.5 & 0.1 & 0 & 76 & 0 \\ 2 & 0 & 0.1 & 0.2 & 2 & 2 & 0 & 2 & 0 & 99 \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \\ x5 \\ x6 \\ x7 \\ x8 \\ x9 \\ x10 \end{bmatrix} \quad \text{y} \quad \boldsymbol{b} = \begin{bmatrix} 100 \\ 100 \\ 100 \\ 100 \\ 100 \\ 100 \\ 100 \\ 100 \\ 100 \\ 100 \end{bmatrix}$$

Etapa 1:

Para el sistema dado se calcularon los pesos de las 10 incógnitas conforme la ecuación (9). El resultado se ilustra en la figura 2

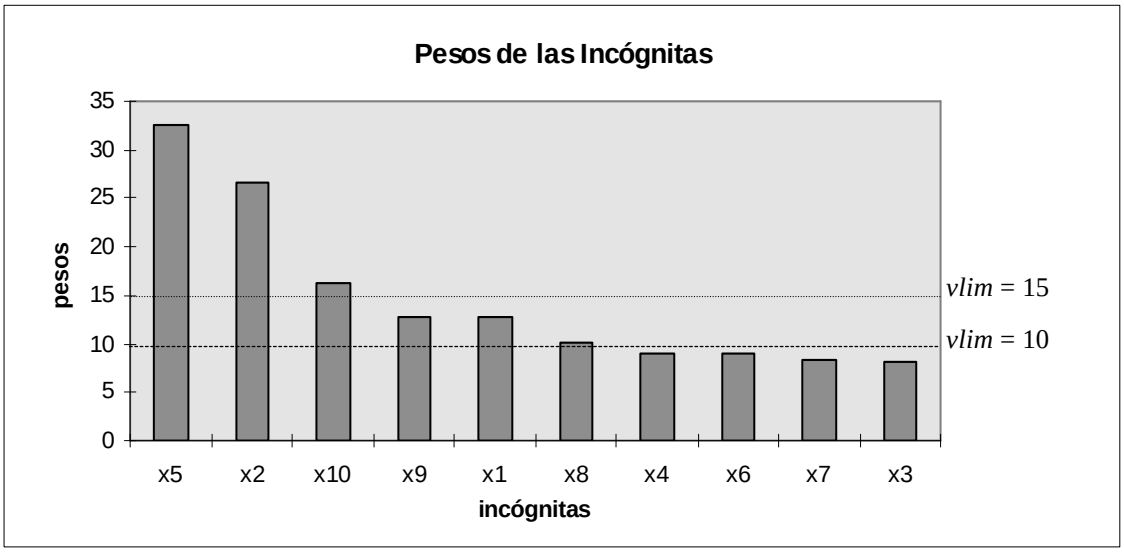


Figura 2: Pesos de la incógnitas del Sistema 10x10/1

Etapa 2:

Se toman los valores:
vlim = 15 y10,
ngrup = 3, 4 y 5,
nviz = 1, 2, 3, 4 y 5

Combinando los valores de *vlim, ngrup* y *nviz* se generan las posibles ternas para la selección de las semillas.

Así para la terna *vlim* = 10 *ngrup* = 3 y *nviz* = 2,
se tiene como conjunto ***K*** = { x_5, x_2, x_{10},x_9, x_1, x_8 } (ver figura 2)
y se genera la tabla 1, en la que colocamos los conjuntos que se van formando para explicar mejor el algoritmo:

	a	b	c	d	e	f
1	***I***	***CIA***	***I***	***CIA***	***I***	***CIA***
2	x_5	x_1 x_2 x_3 x_6 x_7 x_9 x_{10}	x_2	x_1 x_3 x_5 x_7 x_8 x_9	x_{10}	x_1 x_3 x_4 x_5 x_6 x_8
3	x_2	x_8	x_5	x_6 x_{10}	x_5	x_2 x_7 x_9
4	x_{10}	x_4	x_{10}	x_4	x_2	
5	x_9		x_9		x_9	
6	ΣP	88.23	ΣP	88.23	ΣP	88.23

	g	h	i	j	k	l
1	***I***	***CIA***	***I***	***CIA***	***I***	***CIA***
2	x_9	x_1 x_2 x_3 x_4 x_5 x_6 x_7	x_1	x_2 x_4 x_5 x_7 x_8 x_9 x_{10}	x_8	x_1 x_2 x_4 x_6 x_{10}
3	x_5	x_{10}	x_5	x_3 x_6	x_2	x_3 x_5 x_7 x_9
4	x_2	x_8	x_2		x_5	
5	x_{10}		x_{10}		x_{10}	
6	ΣP	88.23	ΣP	88.16	ΣP	85.61

Tabla 1: Cuadro de seleción de semillas del Sistema 10x10/1.

Esta tabla se genera aplicando el algoritmo 2 como sigue;

Las incógnitas x_5, x_2, x_{10}, x_9, x_1 y x_8 (casillas a2, c2, e2, g2 ,y2 y k2) son las candidatas a semillas. En los ***CIA***s se colocan sus incógnitas adyacentes (casillas b2, d2, f2, h2, j2 y l2).

Sucesivamente, se coloca en ***I*** la incógnita de su ***CIA*** que posea el mayor valor de peso, y se la retira del mismo. Una vez completada la tabla (con *ngrup* incógnitas más la semilla), se elige el mayor valor de la sumatoria de los pesos (ΣP). Como en el ejemplo dichas sumatorias son iguales, se toma la primera incógnita (x_5) como primera semilla. Ahora, se verá si la siguiente incógnita (x_2) puede ser otra semilla, para lo cual se debe verificar que dicha incógnita no se encuentre entre las *nviz* incógnitas siguientes a la semilla previamente elegida (x_5). Como *nviz* en este caso vale 2 y la incógnita x_2 está en la posición 1(a3), las incógnita x_2 no puede ser semilla. Lo mismo ocurre con x_{10}, que se

encuentra en la 2da posición(a4). La incógnita x_9 se encuentra en la 3ra posición(a5), por lo cual es la indicada para ser la siguiente semilla

Las semillas elegidas son así: x_5 y x_9

Para las demás ternas se procede análogamente, generándose diversos pares de semillas.

Etapa 3:

Para las semillas x_5 y x_9 se obtiene la partición que se observa en la tabla 2.

	a	b	c	d
	Conjunto 1		Conjunto 2	
1	***I***	***CIA***	***I***	***CIA***
2	x_5	$x_1\ x_2\ x_3\ x_6\ x_7\ x_{10}$	x_9	$x_1\ x_2\ x_3\ x_4\ x_6\ x_7$
3	x_2	x_8	x_3	
4	x_1	x_4		
5	x_{10}			
6	x_8			
7	x_4			
8	x_6			
9	x_7			

Tabla 2

Esta tabla se genera aplicando el algoritmo 3 como sigue:

Se construyen dos conjuntos de incógnitas ubicando las semillas en sus lugares correspondientes (a2, c2) y colocando en los ***CIA***s las incógnitas adyacentes a las mismas.

Como el conjunto 2 ya completó su cupo ($\varphi_2 = 1 > \varphi_1 = 0$) el conjunto 1 incluirá automáticamente a las x_2, x_1 y x_{10}.

La siguiente incógnita candidata a formar parte del subconjunto 1 es x_8, la cual puede ser incluida directamente por no encontrarse en el CIA del subconjunto 2. La siguiente incógnita candidata (x_4) se encuentra en ambas ***CIA***s, es decir es candidata tanto para uno como para otro conjunto. Observamos en la matriz ***A*** que $a_{48} = 1{,}1$ es mayor que $a_{49} = 1$, por lo cual x_4 pasa a formar parte del conjunto 1. El procedimiento continúa hasta que todas las incógnitas hayan sido incluidas por algún conjunto, completándose la partición.

La partición resultante está dada por los conjuntos $\{x_5, x_2, x_1, x_{10}, x_8, x_4, x_6 , x_7\}$ y $\{x_9, x_3\}$

Utilizando los demás conjuntos de semillas generados según la etapa 2 se obtendrán diversas particiones, entre las cuales se seleccionará la mejor en la siguiente etapa.

Etapa 4: Evaluación de particiones

El parámetro de acoplamiento es calculado evaluando la siguiente expresión:

$A = \Sigma |a_{ij}|$ para todo par de incógnitas (x_i, x_j) separadas por la partición.

En el ejemplo: $A = |a_{19}| + |a_{92}| + |a_{94}| + |a_{95}| + |a_{96}| + |a_{97}| + |a_{98}| + |a_{910}| + |a_{31}| + |a_{32}| + |a_{34}| + |a_{35}| + |a_{36}| + |a_{37}| + |a_{38}| + |a_{310}| = 8{,}6$

La partición seleccionada será aquella que posea el menor valor del parámetro A

4. RESULTADOS EXPERIMENTALES

Se presenta aquí el comportamiento experimental para tres problemas tipos denominados aquí:

i. Sistema 10x10/1 para un sistema medianamente acoplado
ii. Sistema 10x10/2 para un sistema débilmente acoplado
iii. Sistema 10x10/3 para un sistema fuertemente acoplado

En el sistema distribuido utilizado, $\boldsymbol{w} = [4\ 1]^T$, es decir que un procesador posee cuatro veces más capacidad de procesamiento que el otro; por ello se divide el cálculo de las 10 incógnitas de los ejemplos en dos subconjuntos de 8 y 2 incógnitas respectivamente.

Se realizó un estudio exhaustivo de todas las posibles formas de particionar en dos grupos de 8 y 2 incógnitas los problemas ejemplos, esto es $C_2^{10} = \frac{10\ !}{(10-2)\ !\ 2\ !} = 45$ formas posibles. Dichas particiones fueron numeradas de 1 a 45. Con cada una de las particiones se resolvieron los sistema y se registraron el número de iteraciones promedio (resultante de pruebas realizadas con seis valores distintos de tolerancia). Se calculó el parámetro de acoplamiento "A" para cada una de las 45 particiones posibles. Las simulaciones de resolución de las 45 particiones con los 6 posibles valores de tolerancia se realizaron en una máquina secuencial.

El llamado "parámetro de acoplamiento A" representa, en valor absoluto, la suma del valor de los coeficientes que se cortan en la partición.

En la tabla 3, se puede observar que las particiones generadas por el método propuesto están en general dentro de las mejores particiones posibles respecto al número de iteraciones y el parámetro de acoplamiento. Esta tabla presenta datos del sistema

10x10/1 (medianamente acoplado). La metodología propuesta genera las particiones nº 23, 7, 2, 39,
18 y 44 como particiones posibles. De estas particiones se elige como primera la partición que posee el menor parámetro de acoplamiento, según la tabla. También se observan el número de iteraciones promedio y las posiciones en el ranking del estudio exhaustivo.

Número de partición	Conjunto 1	Conjunto 2	Iteraciones[+]	Posición en el ranking del número de iteraciones*	Parámetro de acoplamiento
23	$x_1\ x_2\ x_4\ x_5\ x_6\ x_7\ x_8\ x_{10}$	$x_3\ x_9$	4.83	4º	8.6
7	$x_2\ x_3\ x_4\ x_5\ x_6\ x_7 x_9\ x_{10}$	$x_1\ x_8$	4.33	2º	10
2	$x_1\ x_2\ x_3\ x_4\ x_5\ x_6 x_7\ x_9$	$x_8\ x_9$	4.83	4º	10.5
39	$x_2\ x_3\ x_4\ x_5\ x_6 x_7\ x_9 x_{10}$	$x_1\ x_8$	5	5º	10.6
18	$x_1 x_2\ x_3\ x_5\ x_6 x_7\ x_9 x_{10}$	$x_4\ x_8$	6	9º	11.3
44	$x_1 x_2\ x_3\ x_4\ x_5 x_7\ x_8 x_9$	$x_6\ x_{10}$	4.67	3º	12.7

[+] El número de iteraciones es un promedio de pruebas realizadas con seis valores posibles de tolerancia.
* Existen 16 posibles valores del número de iteraciones que van desde 4,17 hasta 7,33.

Tabla 3

En las figuras 3, 4 y 5 se presentan todos los valores del número de iteraciones promedio para todas las particiones posibles de los sistemas i, ii y iii respectivamente.

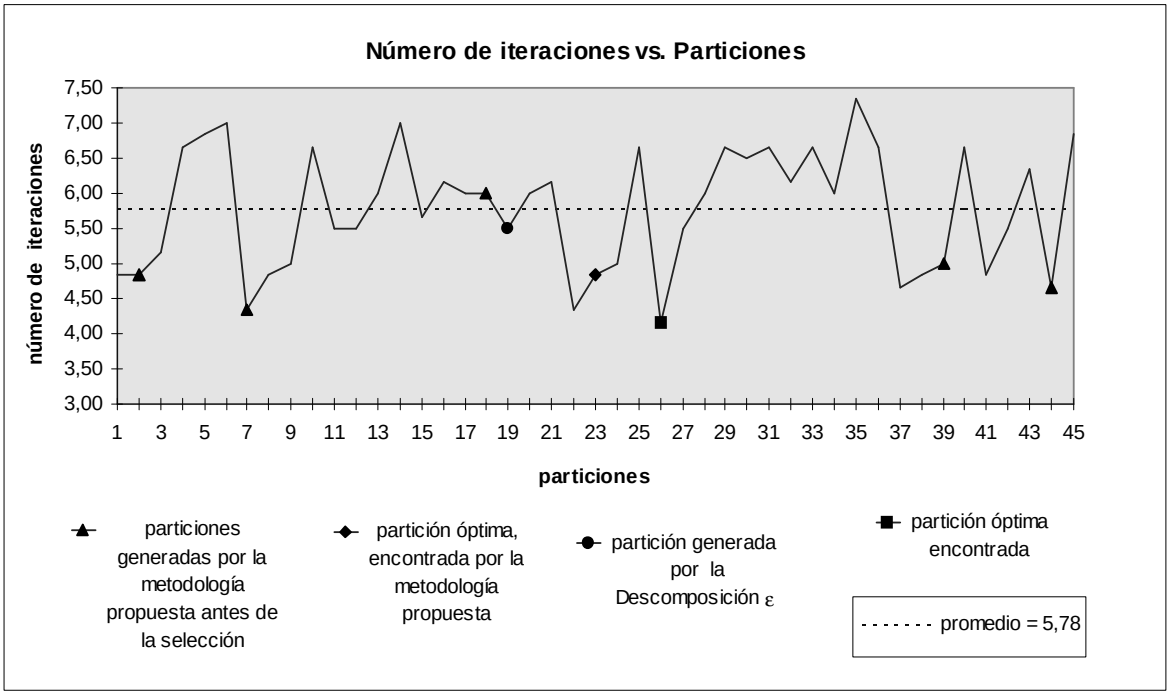


Figura 3: Sistema 10x10/1

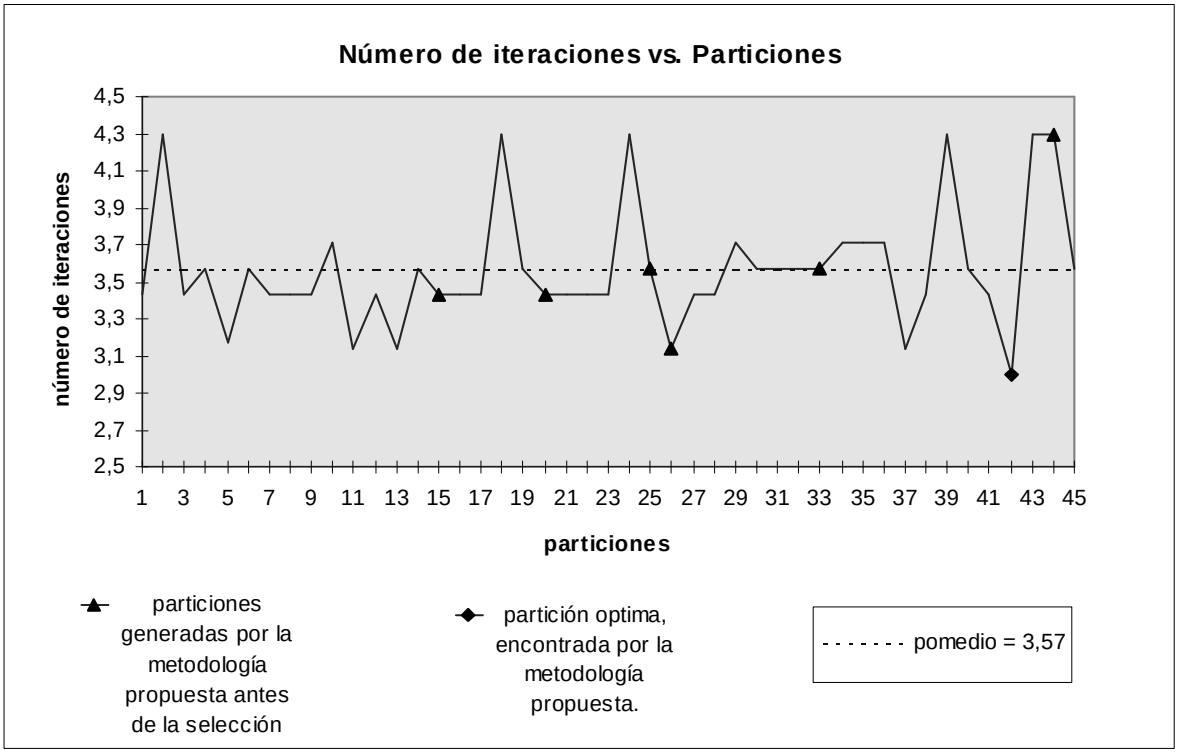


Figura 4: Sistema 10x10/2

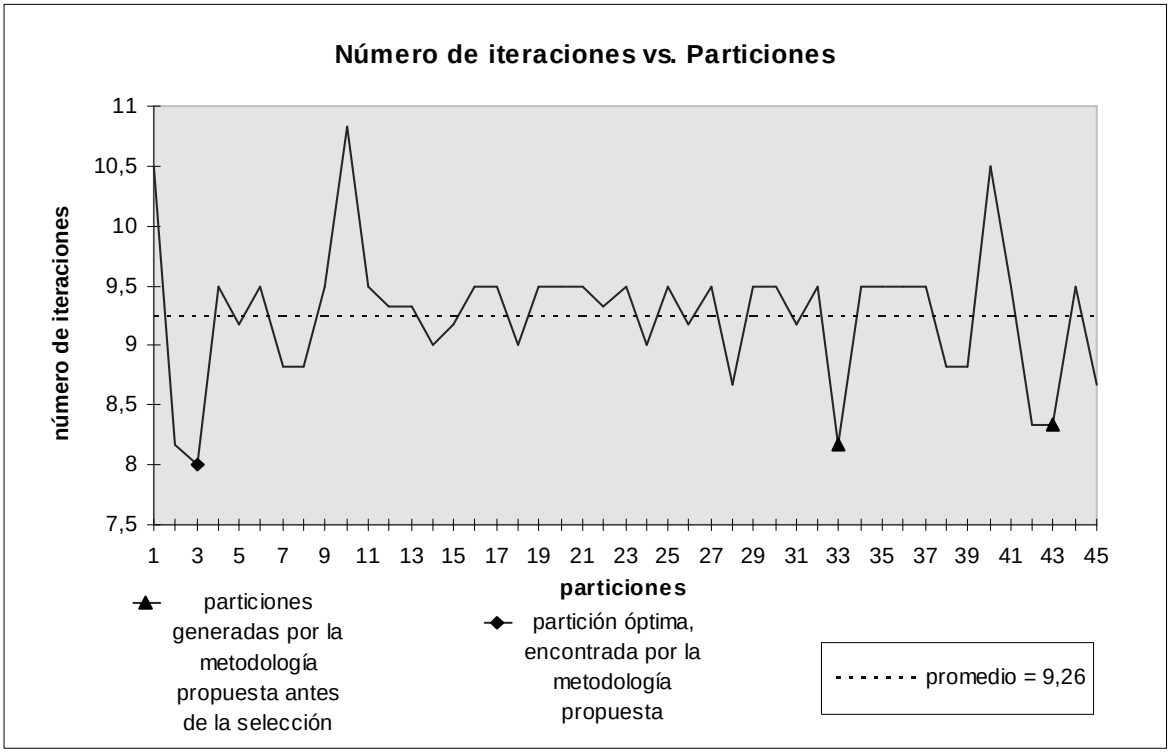
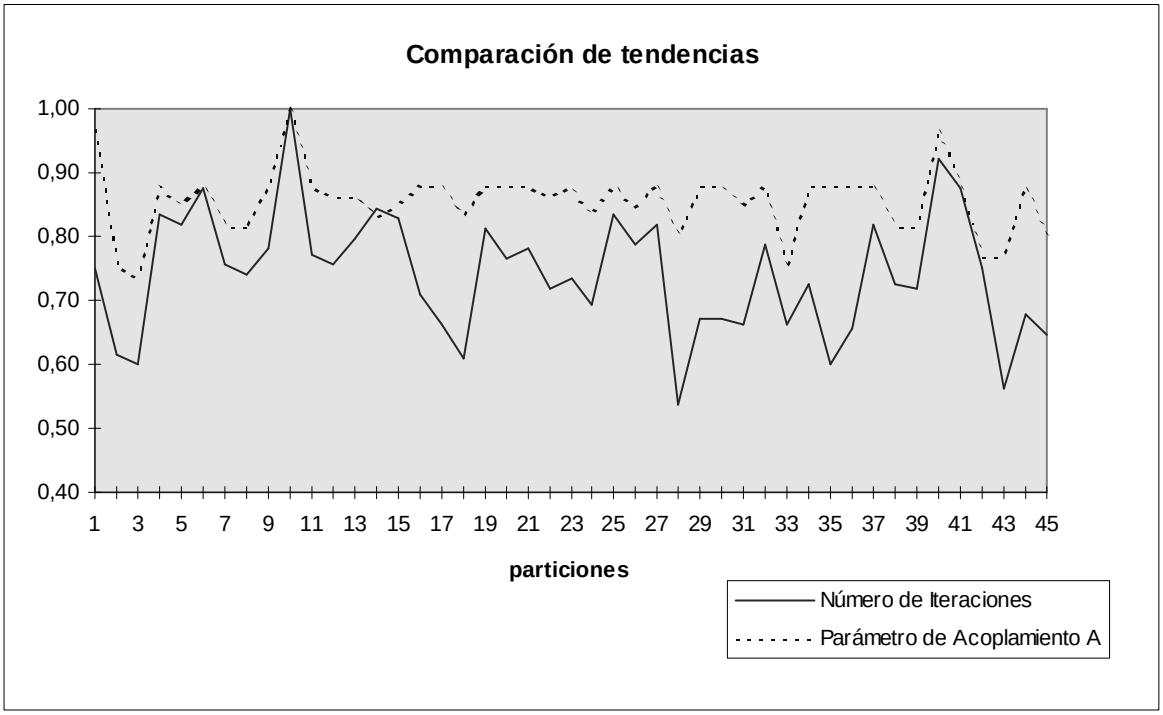


Figura 5: Sistema 10x10/3



Para apreciar mejor la correlación entre las curvas, éstas se han normalizado.

Figura 6: Comparación de tendencias del Sistema 10x10/3

Se observa en la figura 3 que las primeras tres particiones generadas por el método propuesto son mejores que el promedio, y que en general son mejores que la partición resultante de aplicar la descomposición ε.

En la figura 4 se puede observar que la mejor partición generada es la mejor partición posible y que las otras particiones generadas, en general resultan iguales o mejores que el promedio. La descomposición ε no genera ninguna partición adecuada a la performance relativa entre los procesadores.

En la figura 5 se puede observar que la mejor partición generada por la metodología propuesta es la mejor partición posible y que las otras dos particiones generadas resultan muy buenas, ya que están muy cerca de la óptima. Nuevamente, la descomposición ε no genera particiones con el balanceamiento requerido.

El llamado “parámetro de acoplamiento A”, como se dijo anteriormente, representa en valor absoluto, la suma del valor de los coeficientes que se cortan en la partición. Se puede observar en la figura 6 la correlación existente entre el valor de dicho parámetro y el número de iteraciones para el Sistema 10x10/3. Se debe acotar, sin embargo, que dicha correlación puede ser menor en alguno casos. Las bondades y limitaciones del parámetro de acoplamiento como criterio de selección están siendo objeto de estudio y nuevas propuestas están siendo estudiadas.

5 CONCLUSIONES

De los resultados experimentales se puede concluir que:

- La metodología propuesta genera mejores particiones que las generadas en promedio por una elección aleatoria, como se puede observar en las figuras 3, 4 y 5.

- La mejor partición generada por la metodología propuesta es sin duda una buena partición, inclusive en ciertos casos es la mejor partición posible, como se puede observar en las figuras 4 y 5.

- El parámetro de acoplamiento ”A” definido en este trabajo es un buen parámetro para elegir una buena partición, como lo demuestran los resultados experimentales

- La partición generada por Descomposición ε para el sistema 10x10/1 no es mejor que la generada por la metodología propuesta. En los otros casos, la Descomposición ε no genera la dimensión requerida ó la partición generada no está de acuerdo con la performance relativa de los procesadores. Podemos afirmar por lo tanto que la metodología propuesta en este trabajo ofrece mejores resultados que la Descomposición ε.

En resumen, la presente propuesta nos brinda una solución superior a las hoy existentes, a un costo computacional accesible.

6. BIBLIOGRAFÍA

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

[2] D. P. Bersekas, J. N. Tsitsiklis, *Parallel and Distributed Computation. Numerical Methods* Prentice-Hall. 1989.

[3] B. A. Carré, "Solution of Load-Flow by Partitioning Systems into Trees", *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-88, pp. 1931-1938. Noviembre 1968.

[4] G. Feingold, R. S. Varga, "Block Diagonally Dominant Matrices and Generalizations of the Gerschgorin Circle Theorem", *Pacific Journal of Mathematics*, no. 12, pp. 1241-1250. 1962.

[5] M. R. Irving, M. J. H Sterling,"Optical Network Tearing Using Simulated Annealing", *IEEE Proceedings*, vol. 137, no. 1, pp. 69-72. Enero 1990.

[6] M. K. Jain, N. D. Rao, "A Power System Networks Decomposition for Network Solutions", *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-92, no. 2, pp. 619-625. 1973.

[7] E. Kaskurewitz, A. Bhaya, D. D. Šiljak, "On the convergence of parallel asynchronous block-iterative computations", *Linear Algebra Appl.*, 131, pp. 139-160. 1990.

[8] M. H. Mickle, W. G. Vogt, R. G. Colclaser, "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.

[9] Okuguchi, "Matrices with Diagonal Blocks and Economic Theory", *Journal of Mathematical Economics*", no. 5, pp. 43-52. 1978.

[10] F. Pearce, "Matrices with Dominating Diagonal Blocks", *Journal of Economic Theory*, no. 9, pp. 159-170. 1974.

[11] A. O. M. Saheh, M. A. Laughton,"Cluster Analysis of Power System Networks for Array Processing Solutions", *IEEE Proceedings*, vol. 132, no. 4, pp. 172-178. July 1985.

[12] A. M. Sasson, "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.

[13] M. Sezer, D. D. Šiljak, “Nested epsilon decompositions of complex systems. IFAC” 9th *World Congress*, Budapest, Hungría.

[14] M. Sezer, D. D. Šiljak, “Nested epsilon decompositions and clustering of complex systems”, *Automática*, vol. 22, no. 3, pp. 69-72. 1986.

[15] M. Sezer, D. D. Šiljak, “Nested epsilon decompositions of linear systems: Weakly coupled and overlapping blocks”, *SIAM Journal of Matrix Analysis and Aplications*, 12, pp. 521-533. 1991.

[16] M. Sezer, D. D. Šiljak, “Epsilon decompositions of linear systems: Weakly coupled and overlapping blocks”, *research supported by National Science Foundation* - ECS-8315327.

[17] J. M. Undrill. H. H. Happ, “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.

[18] M. H. M. Vale., D. M. Falcão, E. Kaszkurewicz., “Electrical Power Network Decomposition for Parallel Computations”. *IEEE Iternational Symposium on Circuits and Systems* - ISCAS 92. San Diego, California. 1992.

[19] A. Y. Zecevic, D. D. Siljak, “Balanced Decompositions of Sparse Systems for Parallel Processing”, *IEEE Trasactions on Circuits and Systems*, vol. 41, no. 3, pp. 220-233. Marzo 1994.