Universidad Nacional del Este.
Facultad Politécnica.

Computación paralela para ajuste coordinado de controladores de Sistemas Eléctricos.

Por: **María Lorena Giménez Torres.**
y **Lilian María Morel Decoud.**

Año 2017.

Universidad Nacional del Este.
Facultad Politécnica.

Carrera Ingeniería de Sistemas.
Cátedra Trabajo Final de Grado.

Computación paralela para ajuste coordinado de controladores de Sistemas Eléctricos.

Por: **María Lorena Giménez Torres.**
y **Lilian Maria Morel Decoud.**

Profesor Orientador: **Ing. D.Sc. Enrique Ramón Chaparro Viveros.**
Profesor Co-orientador: **Ing. M.Sc. Eustaquio Alcides Martínez.**

Trabajo final de grado presentado a la Facultad Politécnica de la Universidad Nacional del Este como parte de los requisitos para optar al título Ingeniera de Sistemas.

Ciudad del Este, Alto Paraná. Paraguay.

11 de octubre de 2017

FICHA CATALOGRÁFICA
BIBLIOTECA DE LA FACULTAD POLITÉCNICA
DE LA UNIVERSIDAD NACIONAL DEL ESTE

Giménez Torres, María Lorena, 1991.
Morel Decoud, Lilian María, 1991
Computación paralela para ajuste coordinado
de controladores de Sistemas Eléctricos.
Ciudad del Este, Alto Paraná. Año: 2017.
Páginas: 77.

Orientador: Ing. D.Sc. Enrique Ramón Chaparro.
Área de estudio: Algoritmos Genéticos, Programación Paralela.

Co-Orientador: Ing. M.Sc. Eustaquio Alcides Martínez
Área de estudio: Algoritmos Genéticos, Programación Paralela.
Carrera: Ingeniería de Sistemas.
Titulación: Ingeniera de Sistemas.
Trabajo Final de Grado. Universidad Nacional del Este,
Facultad Politécnica.

Palabras claves: 1. Programación Paralela, 2. Algoritmos Genéticos,
3. Comunicación Síncrona.
Parallel Computing for Coordinated Tuning of Electrical Systems Controllers.
Key words: 1. Parallel Programming, 2. Genetic Algorithms,
3. Synchronous Communication.

Yo, D.Sc. Ing. Enrique Ramón Chaparro, documento de identidad No. 965.527, Profesor Orientador del TFG titulado "Computación paralela para ajuste coordinado de controladores de Sistemas Eléctricos.", de las Alumnas María Lorena Giménez Torres, documento de identidad No. 3.190.781 y Lilian María Morel Decoud, documento de identidad No. 3.396.298 de la carrera Ingeniería de Sistemas de la Facultad Politécnica de la Universidad Nacional del Este; certifico que el mencionado Trabajo Final de Grado ha sido realizado por dichas Alumnas, de lo cual doy fe y en mi opinión reúnen las condiciones para su presentación y defensa ante la Mesa Examinadora designada por la institución.

Ciudad del Este, _____ de _____ de 2017

_____

Ing. D.Sc. Enrique Ramón Chaparro

Nosotros, los miembros de la Mesa Examinadora del Trabajo Final de Grado titulado "Computación paralela para ajuste coordinado de controladores de Sistemas Eléctricos.", de la carrera Ingeniería de Sistemas de la Facultad Politécnica de la Universidad Nacional del Este, hacemos constar que el citado trabajo ha sido evaluado en fondo y forma por esta Mesa, la que por _____ ha resuelto asignar la calificación _____

Ciudad del Este, _____ de _____ de 2017

_____

Profesor _____

Presidente de la Mesa Examinadora

_____ _____

Profesor _____ Profesor _____

Miembro de la Mesa Examinadora Miembro de la Mesa Examinadora

Este logro es para mis padres quienes fueron siempre mi mayor ejemplo de amor, sacrificio y superación. Por impulsarme siempre a luchar por mis objetivos.

Lilian Morel.

A mis Padres.
Lorena Giménez.

Agradecida a Dios por su bendición constante, a mis padres por la educación, paciencia y su apoyo incondicional, a mi hermana por la confianza y el apoyo de siempre. A mi esposo por su optimismo que siempre me impulso a seguir adelante.
También quiero agradecer a nuestro tutor el Ing. D.Sc. Enrique Ramon Chaparro Viveros por su entera disposición y su paciencia. Al co-tutor el Ing. M.Sc. Eustaquio Alcides Martínez por sus recomendaciones, su ayuda y su total disposición siempre.
Al Lic. Luis Moran por su ayuda incondicional en muchas ocasiones.
A mi compañera de tesis Lorena Giménez por su apoyo constante y compañerismo.
Lilian Morel.

A mis padres, porque todo esto fue posible gracias a su apoyo incondicional.
A nuestro tutor, Dr. Enrique Chaparro, por la orientación y ayuda que nos brindó en la realización de este proyecto. También a nuestro co-tutor, quien nos brindó sus conocimientos y ayuda que necesitamos.
Al Lic. Luis Morán, porque sin su ayuda no habríamos logrado un avance rápido del proyecto.
A mi compañera de tesis, por todo.
Lorena Giménez.

Resumen

Este trabajo presenta una aplicación de Programación Paralela para los Algoritmos Genéticos (AG) al caso concreto de problemas de Ajuste coordinado de controladores de Sistemas Eléctricos. Al AG se aplica la computación paralela del tipo Maestro − Esclavo y del tipo Multipoblacional de comunicación síncrona mediante MPI los cuales fueron desarrollados en la plataforma libre llamada PelicanHPC, en el lenguaje de programación GNU Octave.

Ambos algoritmos fueron ejecutados en un cluster de computadoras interconectadas a través de una red de comunicación Ethernet, luego se comparan los resultados entre el AG secuencial, el AGP del tipo Maestro - Esclavo y del tipo Multipoblacional.

Con la implementación de la computación paralela al AG se logran resultados satisfactorios tanto desde el punto de vista de la calidad de resultados obtenidos en cuanto al Fitness como desde el punto de vista de mejora de la eficiencia computacional alcanzada por las versiones paralelas de los algoritmos con respecto a su contraparte secuencial.

Palabras claves: 1. Programación Paralela, 2. Algoritmos Genéticos, 3. Comunicación Síncrona.

Abstract

This work presents an application of Parallel Programming for Genetic Algorithms (GA) to the concrete case of problems of coordinated adjustment of Electrical Systems controllers. The GA applies the parallel computation of the Master - Slave type and the Multipoblational type of synchronous communication through MPI which were developed in the free platform called PelicanHPC, in the programming language GNU Octave.

Both algorithms were executed in a cluster of computers interconnected through an Ethernet switch network, then the results were compared between the sequential GA, the Master - Slave type AGP and the Multipoblacional type.

With the implementation of parallel computing to GA, satisfactory results are obtained both from the point of view of the quality of results obtained in Fitness and from the point of view of improving the computational efficiency achieved by the parallel versions of the algorithms with with respect to its sequential counterpart.

Key words: 1. Parallel Programming, 2. Genetic Algorithms, 3. Synchronous Communication.

Índice general

Índice de figuras

Índice de Tablas

Acrónimos

SEP Sistemas Eléctricos de Potencia. 1, 2 , 24, 26 , 29 , 31, 32 , 34, 35 , 37, 38 , 45, 46 , 53, 54

MPI Message Passing Interface. 1 , 17 , 32, 36 , 53, 54

RAT Reguladores Automáticos de Tensión. 23 , 25, 28 , 31, 33 , 35 , 37, 38

ESP Estabilizadores de Sistemas de Potencia. 23, 26 , 31, 35

AG Algoritmo Genético. 1 , 17 , 32, 36 , 53, 54

LAN Local Area Network. 6

AGP Algoritmo Genético Paralelo. 1 , 53

Capítulo 1

Introducción

El problema de ajuste coordinado de controladores del SEP [3] requieren del análisis de un gran número de variables complejas y por ende demandan un amplio uso de recursos computacionales. Este trabajo propone un algoritmo genético paralelo para resolver el problema anteriormente mencionado. Los algoritmos genéticos son métodos de búsqueda basados en los principios de la selección natural y la genética [4] [5]. En muchas aplicaciones prácticas los AGs encuentran una buena solución en un lapso de tiempo considerable [4] [5] . Hay muchos estudios para mejorar el rendimiento de los AGs y una de las alternativas más interesantes es el uso de las implementaciones paralelas. El éxito de los AGPs reside en la reducción del tiempo requerido para encontrar una solución aceptable en muchos problemas complejos [4]. Para este proyecto se han desarrollado dos tipos de AGPs: Maestro − Esclavo y multipoblacional. La comunicación entre procesos se realiza a través de MPI [13] y se ejecuta en un cluster de hasta 6 procesos en paralelo.

1.1. Motivación

Muchos problemas de optimización que aparecen en el ámbito de las ingenierías son muy difíciles de solucionar por medio de técnicas tradicionales, por lo que a menudo se aplican algoritmos evolutivos, inspirados en la naturaleza, que recogen un conjunto de modelos basados en la evolución de los seres vivos. Haciendo uso de la computación paralela, un algoritmo genético debe ser capaz de encontrar mejores soluciones en un tiempo notablemente reducido.

1.2. Definición del problema

La necesidad de ejecución de un algoritmo de optimización multi-objetivo, analizando varios escenarios, con parámetros adecuados de ajuste de controladores del SEP, con tiempos computacionales reducidos y menores requerimientos computacionales, en comparación con técnicas clásicas actuales.

1.3. Objetivos

1.3.1. Objetivo general

Implementar computación paralela para ajuste coordinado de controladores de Sistemas Eléctricos para la obtención de mejores resultados en tiempo reducido.

1.3.2. Objetivos específicos

- Implementar computación paralela síncrona del tipo Maestro - Esclavo al algoritmo genético meta-heurístico de ajuste coordinado de controladores de sistemas eléctricos secuencial.
- Implementar computación paralela síncrona del tipo Multipoblacional al algoritmo genético meta-heurístico de ajuste coordinado de controladores de sistemas eléctricos secuencial.
- Validar las implementaciones computacionales de los algoritmos genéticos paralelizados del tipo Maestro - Esclavo y Multipoblacional mediante ejecuciones en un cluster.
- Comparar los resultados obtenidos del algoritmo genético secuencial con las implementaciones computacionales paralelas del tipo Maestro - Esclavo y Multipoblacional ejecutados en un cluster de computadoras.

1.4. Hipótesis

El algoritmo genético paralelo del Ajuste Coordinado de Controladores del SEP posibilita la reducción del tiempo de ejecución a bajo costo de implementación y mejora los resultados.

1.5. Justificación

El problema de ajuste coordinado de controladores de sistemas eléctricos de gran porte requiere de un tiempo excesivo de ejecución y gran poder de cómputo, para alivianar la carga computacional del mencionado problema se implementa la computación paralela junto con librerías de paso de mensajes y se utiliza un cluster para su ejecución. Cuando una aplicación paralela es ejecutada en un cluster, su código y los datos son distribuidos entre los procesadores para su procesamiento con el fin de obtener un buen desempeño y mejores resultados.

1.6. Delimitación del trabajo.

- Aplicación de computación paralela a un algoritmo genético.
- Cálculo del speedup en base a los resultados obtenidos.
- Cálculo de la eficiencia a partir de los resultados del speedup.
- Comparación de los resultados de todas las ejecuciones realizadas.

1.7. Descripción de los contenidos por capítulo.

El contenido de este proyecto consta de 5 capítulos los cuáles se describen brevemente a continuación:

- Capítulo I: Describe los detalles más resaltantes del proyecto, ampliando algunos puntos como la motivación, definición del problema, el objetivo general, los objetivos específicos, hipótesis, la justificación y así también la delimitación del alcance del proyecto.
- Capítulo II: Se presentan conceptos necesarios para una mayor comprensión acerca de Computación Paralela y Algoritmos Genéticos, así también se presentan las herramientas utilizadas para la realización del proyecto.
- Capítulo III: Expone los conceptos fundamentales que se requiere para una mayor comprensión acerca de Control de Sistemas Eléctricos de Potencia.

- Capítulo IV: Se detalla la metodología utilizada en el proyecto, se mencionan también algunos pasos que fueron necesarios para la adaptación de los algoritmos mencionados y se describe el diseño del modelado para la implementación de los diferentes tipos de algoritmos paralelos que fueron utilizados.
- Capítulo V: Se presentan los resultados obtenidos de las pruebas realizadas, así también gráficos que denotan dichos resultados.
- Capítulo VI: Se evalúan las implicaciones y se exponen las conclusiones a las que se han llegado en base a los resultados obtenidos, los logros obtenidos y algunas sugerencias para futuras investigaciones.

Capítulo 2

Computación Paralela y Algoritmos Genéticos

2.1. Computación Paralela

La computación paralela es una forma de cómputo en la que muchas instrucciones se ejecutan simultáneamente operando sobre el principio de que problemas grandes, a menudo se pueden dividir en unos más pequeños que luego son resueltos simultáneamente en varios procesadores [21].

2.1.1. Cómputo paralelo en cluster

El cómputo paralelo es la ejecución simultánea de alguna tarea en varios procesadores con el fin de reducir el tiempo de ejecución de un algoritmo. Para disminuir el tiempo de ejecución del algoritmo, código y/o los datos de la aplicación, son divididos y distribuidos entre los procesadores. Cada procesador ejecuta una o más tareas que realizan una parte del cómputo del algoritmo. Los procesadores son interconectados por un bus o una red de conmutación que permite a las tareas compartir datos o sus resultados. El cómputo en paralelo ha sido ampliamente utilizado para mejorar el desempeño de modelos que demandan gran cantidad de cómputo. Algunos de éstos modelos en general se caracterizan porque realizan gran cantidad de operaciones de cálculo en la solución de problemas de ciencias e ingeniería [21].

Elementos de un cluster

En los clusters, cada computadora está conformada por una placa madre, unidad central de procesamiento (CPU) o procesador, memoria y adaptador de red donde se procesan los datos de un modelo dado. Existen dos tipos de

clusters [21]: homogéneo y heterogéneo. En un homogéneo las computadoras tienen las mismas características como la velocidad de procesador, el tamaño de la memoria, la velocidad del adaptador del reloj, entre otras. Otro elemento del cluster es el software especializado, que ayudará a distribuir el trabajo entre las computadoras. Para distribuir el trabajo entre las computadoras se requiere que se interconecten entre sí, para que se puedan comunicar unas con otras.

- CPU: la CPU o procesador de propósito general, realiza trabajos de oficina, casa o ingeniería que cuenta con al menos un núcleo. Los procesadores o núcleos contienen varias formas de paralelismo. La principal y mas simple esta en el uso de palabras de gran tamaño. Para describirla por ejemplo en la clase SIMD que pertenece a la arquitectura Flynn se puede mencionar MMX, 3DNow, SSE, AltiVec las cuales permiten realizar la misma operación en un grupo de números. Así que, la CPU usa paralelismo en la ejecución de una instrucción. La ejecución segmentada del estado de la instrucción es otro ejemplo de paralelismo [21].

- Interconexión: actualmente las LANs ayudan a compartir recursos, facilitando la comunicación entre procesadores dentro de una institución, organización u hogar. Las LANs abarcan áreas relativamente pequeñas, por lo general se emplean en edificios donde pueden conectarse con las otras LANs mediante el Internet. Por su gran demanda son cada vez mas rápidas y mas baratas, lo cual ha permitido que sean una exelente alternativa para ejecutar aplicaciones paralelas. Las computadoras en una LAN son interconectadas por un medio de transmisión para comunicarse. Las LANs tradicionalmente operan con medio de transmisión reales como cables de par trenzado, cables coaxial, fibra óptica, radio y microondas de frecuencia no comerciales. Las computadoras en una LAN son conectadas internamente a enlaces, que ayudan en la comunicación entre sí y evitan pérdida de datos así como colisiones entre los datos. Las interconexiones entre las computadoras para coordinar, sincronizar o intercambiar datos se puede realizar a través de paso de mensajes sobre el enlace, cuando se emplea en un sistema multicomputadora. La tarea principal de interconexión de la red es transmitir mensajes desde un nodo fuente a otro nodo destino. El mensaje puede contener datos o una petición a memoria. El requerimiento en la red es realizar la transferencia correcta del mensaje lo más rápido posible, aún si varios mensajes han sido transmitidos al mismo tiempo.

- Software: hasta ahora se ha definido la infraestructura o el hardware necesario para hacer cómputo paralelo. Como se mencionó anteriormente,

los clusters ofrecen una excelente relación costo-rendimiento. Sin embargo, esto no es suficiente para hacer cómputo paralelo, es necesario además contar con cierto software como un sistema operativo que pueda manejar la computadora paralela y alguna metodología o modelo para desarrollar las aplicaciones paralelas. En cuanto a los sistemas operativos, es común que la arquitectura paralela tenga su propio sistema operativo como Solaris o AIX, y ofrezcan sus propios programas para desarrollo. Algunos de estos programas son buenos, pero la restricción es que solo pueden ser usados si se adquiere la costosa computadora paralela que ellos venden. Si estos programas se logran instalar en arquitecturas tipo cluster, estos podrían ser incompatibles o poco eficientes. La otra opción es usar algún sistema operativo como Linux, que son de distribución libre y gratuitos. Algunos ejemplos de estos sistemas son: Debian y Fedora, entre otros. Por ser de uso general, estos sistemas operativos resultan ser eficientes para máquinas paralelas como lo son los clusters. Adicionalmente existen compiladores y librerías que proveen un mecanismo que permite coordinar el cómputo e intercambio de datos entre las tareas ejecutándose en el cluster. Las librerías más comunes son Message Passing Interface (MPI) y Parallel Virtual Machine PVM.

2.1.2. Modelos de programación basado en el flujo de instrucciones y el número de datos

Las computadoras paralelas pueden ser divididas en dos categorías [21] flujo de instrucciones y datos. La primera está basada en los principios de la computadora de Von Neumann, excepto que múltiples instrucciones pueden ser ejecutadas en algún tiempo dado. El flujo de datos no tiene un registro para activar las instrucciones o un lugar de control. El control es totalmente distribuido donde se lleva a cabo la activación de la instrucción por medio de un evento [21]. La clasificación más popular de arquitecturas de computadoras paralelas es la de Flynn publicada en 1966 [22]. Esta taxonomía de las arquitecturas está basada en la clasificación del flujo de datos e instrucciones en un sistema. Con estas consideraciones, Flynn clasifica los sistemas en cuatro categorías [22]

- Single Instruction, Single Data (SISD). Conocidas como computadoras de series escalares, proveniente de la arquitectura de Von Neumann. En ella se ejecuta una sola instrucción y cuenta con un registro que se denomina contador de programa. Dicho contador se utiliza para la ejecución secuencial del programa. Las instrucciones se almacenan en

la memoria y el contador de programa apunta a la siguiente instrucción para su ejecución. (Fig.2.1)

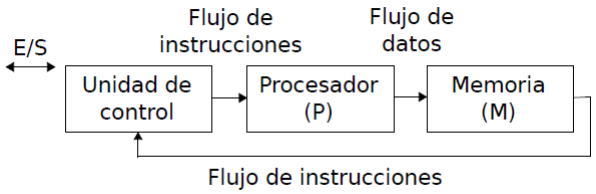

Figura 2.1: Arquitectura SISD.

- Multiple Instruction, Single Data (MISD). Los procesadores vectoriales son clasificados con frecuencia en este modelo. Cada procesador tiene una memoria privada de programa, pero solo un acceso a la memoria global de datos. En cada paso, cada unidad de procesamiento obtiene los mismo datos de la memoria de datos y carga una instrucción de la memoria del programa. Estas instrucciones posiblemente diferentes son ejecutadas en paralelo. Según Fayez como ejemplo de este tipo de arquitectura paralela podemos mencionar a las redes neuronales y las máquinas de flujo de datos.

- Single Instruction, Multiple Data (SIMD). Esta arquitectura manipula una sola instrucción sobre un grupo de datos diferentes al mismo tiempo. Varios procesadores son manipulados por una única unidad de control que es la que distribuye el trabajo a los procesadores y espera, en todo caso el resultado de la ejecución. Al igual que las MISD, las SIMD soportan procesamiento vectorial (matricial) asignando cada elemento del vector a una unidad funcional para el procesamiento concurrente. Es considerado el método más simple de paralelismo y hoy en día es el más común. Las aplicaciones más comunes para esta arquitectura generalmente se utilizan en aplicaciones científicas y de ingeniería. (Fig. 2.2)

- Multiple Instruction, Multiple Data (MIMD). Son consideradas arquitecturas complejas, sin embargo potencialmente ofrecen una mayor eficiencia en las operaciones concurrentes. El trabajo concurrente se realiza cuando se cuenta con varios procesadores para la ejecución operando simultáneamente y además hay varios programas (procesos) ejecutándose también al mismo tiempo. (Fig. 2.3)

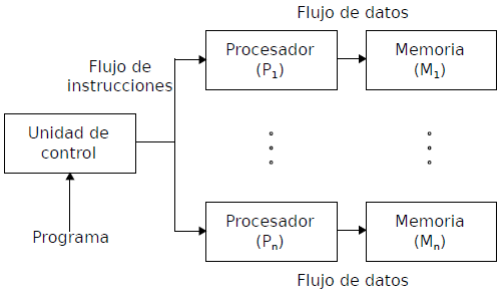

Figura 2.2: Arquitectura SIMD.

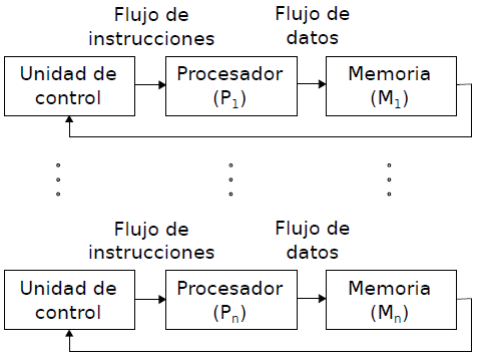

Figura 2.3: Arquitectura MIMD.

El modelo SPMD (del ingles: Simple Program Multiple Data) es el más usado para desarrollar algoritmos paralelos en clusters de computadoras. En este modelo el mismo código del programa dealgoritmo es ejecutado en todos los procesadores. Los datos del algoritmo son divididos y distribuidos en los procesadores donde fue ejecutado el programa. Así, cada procesador ejecuta básicamente la misma pieza de código, pero sobre una parte diferente de los datos. El particionamiento de los datos en este modelo también es conocido como paralelismo geométrico (debido a que los datos generalmente son divididos en bloques del mismo tamaño), descomposición en el dominio, o paralelismo en datos. El proceso de un modelo paralelo generalmente requieren compartir los datos entre ellos, por lo que deben comunicarse y sincronizarse para acceder a los datos. Para realizar la comunicación entre los procesos de un cluster comúnmente se usa el ambiente de programación de paso de mensajes. Este ambiente de programación está basado en una biblioteca que se ejecuta en el espacio de dirección del proceso. Además, la biblioteca provee un conjunto de funciones o primitivas que permiten al programador crear algoritmos paralelos.

2.1.3. Programación con paso de mensajes

Actualmente, el modelo de programación predominante para el cómputo paralelo en clusters con memoria distribuida es el paso de mensajes **??**. Como se recordará, en un cluster los nodos tienen su memoria local y ninguno puede leer o escribir en la memoria de otro; no existe una memoria global accesible para todos los procesos que ejecutan el algoritmo. El intercambio de los datos se realiza haciendo una copia (usando mensajes de envío y recepción), para transferir datos (Fig. 2.4), por ejemplo: desde la memoria local del proceso A a la memoria local del proceso B, A debe enviar un mensaje conteniendo el dato de B, B debe recibir el dato en un buffer de su memoria local. Un mensaje puede ser una instrucción, dato, sincronización o una señal de interrupción. Esta forma de comunicación es la más simple del envío de un mensaje en un modelo paralelo que comúnmente se conoce como punto a punto. Por otro lado, si una algoritmo es divida en procesos; cada uno es ejecutado en procesadores o núcleos distintos. Si el número de procesos es más grande que el número de procesadores o núcleos entonces un proceso compartirá tiempo de ejecución con otro.

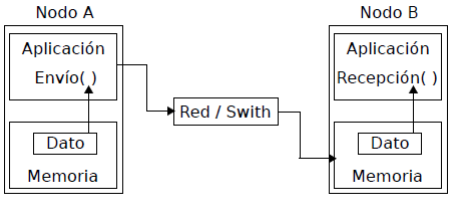

Figura 2.4: Modelo de paso de mensajes.

La comunicación punto a punto está compuesta de dos operaciones: una transmisión y una recepción [24]. La operación de transmisión puede ser síncrona o asíncrona. La transmisión síncrona (envío bloqueante) termina solamente cuando el receptor emite un mensaje que indica que está listo para la recepción. El emisor termina la función de envío y continua con su ejecución, mientras los datos son transmitidos a bajo nivel. El proceso destino iniciará la recepción siempre y cuando tenga espacio suficiente en el buffer de recepción, si no tiene espacio suficiente no emitirá el mensaje que indica que está listo y por lo tanto permanecerá bloqueado (Fig. 2.5). La transmisión asíncrona (envío no bloqueante) termina tan pronto como el mensaje correspondiente se haya entregado al sistema de comunicación. El emisor sigue su ejecución aun cuando no se haya emitido un mensaje por parte del receptor indicando que está listo. Además de este tipo de comunicación, existe la comunicación

colectiva que facilita la transmisión entre procesos. Muchos de los sistemas de paso de mensajes proporcionan operaciones colectivas que permiten que más de dos nodos se comuniquen entre sí. Las operaciones colectivas pueden ser escritas por el programador usando las operaciones básicas de punto a punto. Una transferencia colectiva involucra a más de un emisor o a más de un receptor. De esta manera, un mensaje puede ser transmitido desde un solo proceso a varios procesos, un proceso puede recolectar un conjunto de datos al recibir mensajes de varios procesos para formar uno solo.

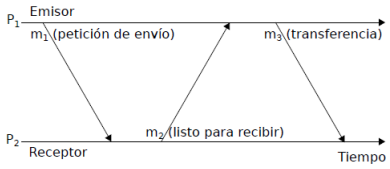

Figura 2.5: Protocolo de envío y recepción bloqueante.

A partir de las operaciones colectivas pueden formarse otras operaciones dependiendo de las necesidades que se tengan. Las comunicaciones punto a punto y las colectivas puede emplearse en una gran variedad de ambientes de programación paralela. Los ambientes de programación paralela proporcionan una biblioteca de comunicación de paso de mensajes que manipulan componentes a bajo nivel de una forma muy transparente para el desarrollador. El desarrollador no se tiene que preocupar por el uso de sockets para transmitir mensajes entre los procesos ya que esta forma de transmisión no es muy adecuada debido a que los sockets son de un nivel de abstracción muy bajo.

2.1.4. MPI

MPI [13] es un estándar para paso de mensajes que ha sido desarrollado por un comité integrado por representantes de laboratorios de investigación, universidades e industriales. El estándar de MPI define la interfaz de usuario y la funcionalidad para un amplio rango de capacidades de paso de mensajes. Desde su primera versión MPI-1 en 1994 donde se definen las operaciones de comunicación estándar. MPI se convirtió en un estándar reconocido y el sistema de paso de mensajes la elección de muchos desarrolladores. Esto se debe tanto al rendimiento demostrado, como a su extensa colección de rutinas para soportar los patrones de comunicación ocurridos comúnmente. La segunda versión MPI-2 en 1998, agregan varias capacidades al estándar base, las más notables son: el control dinámico de procesos y entrada/salida com-

partida. El principal objetivo de MPI, como la mayoría de los estándares, es un grado de portabilidad a través de distintas computadoras. Esto significa que el mismo código fuente puede ser ejecutado en una gran variedad de computadoras siempre que la biblioteca de MPI esté disponible. Otro tipo de compatibilidad ofrecida por MPI, es la capacidad de ejecutarse transparentemente en sistemas heterogéneos, donde el usuario no necesita preocuparse si el programa está enviando mensajes entre procesadores de distintas o similares arquitecturas. La implementación de MPI tiene una amplia biblioteca con las funciones adecuadas que automáticamente realizará lo necesario para efectuar la conversión y utilizará el correcto protocolo de comunicación. MPI consiste en una biblioteca que provee un conjunto de funciones para especificar el paso de mensajes entre procesos. Un requerimiento importante en todos los sistemas de paso de mensajes es garantizar una comunicación segura donde los mensajes no relacionados sean separados de otros. MPI define el concepto de comunicadores el cual combina contextos de mensajes y grupos de tareas para proporcionar mensajes seguros. Los comunicadores pueden ser clasificados en intra-comunicadores para operaciones dentro de un grupo de tareas (Fig. 2.6) e inter-comunicadores para operaciones entre diferentes grupos de tareas. Donde se pueden realizar diferentes tipos de comunicaciones.Para ello se establecen algunos párametros [13]

- Comunicadores: Un comunicador corresponde a un grupo de procesos sobre el que se realiza la comunicación. Dentro de un comunicador cada proceso tiene un rango que lo identifica. MPI_COMM_WORLD es un comunicador. Todas las llamadas MPI para establecer comunicaciones requieren un argumento comunicador y los procesos MPI solo pueden comunicarse si comparten un comunicador.

- Funciones Básicas de Comunicación: La forma de comunicación en MPI es a través de mensajes que contienen datos. La forma más simple es la comunicación punto a punto, donde se envía un mensaje de un proceso a otro. Esto se realiza usando las funciones MPI_Send y MPI_Recv.

Tipos de comunicaciones

MPI provee distintas primitivas de comunicación de punto a punto, bloqueantes y no bloqueantes y tipos de datos derivados. También provee diferentes rutinas de comunicación colectiva entre tareas pertenecientes a un grupo. Otras funciones incluyen manejo de topologías de procesos, funciones de chequeo y control de estado del medio ambiente [21].

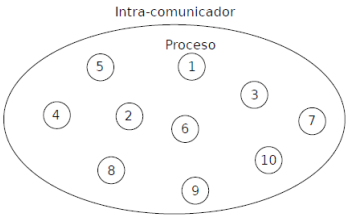

Figura 2.6: Esquema de intra-comunicador.

- Comunicación punto a punto:

  El mecanismo de comunicación básico de MPI es la transmisión de datos entre un par de procesadores, de un lado el emisor y del otro el receptor, comúnmente llamada comunicación punto a punto. En la comunicación punto a punto es posible transmitir valores escalares y arreglos continuos de valores para tipos de datos definidos por MPI. Además, MPI provee un método para el desarrollador, en el cual es posible definir un patrón de datos no continuos y usarlo en funciones de MPI. Estos patrones son llamados tipos derivados y pueden ser creados por una serie de funciones para formar virtualmente patrones arbitrarios de datos y combinación de tipos de datos. Además, la comunicación punto a punto es la base para la creación de otros tipos de comunicaciones que ofrece MPI. La operación de envío en el emisor es realizada por la función MPI_Send donde se especifican los datos a transmitir, el proceso destino que recibirá los datos, una etiqueta para identificar cada mensaje y por último, un comunicador el cual pertenece a un grupo de procesos. La sintaxis es la siguiente:

  MPI_Send(dato, destino, etiqueta, comunicador)

  La función de recepción cuenta con los siguientes parámetros: el emisor, una etiqueta para identificar cada mensaje, y por último el comunicador información del estado de la transmisión. La sintaxis es la siguiente:

  MPI_Recv(emisor, etiqueta, comunicador)

  Las funciones MPI_Send y MPI_Recv son operaciones síncronas bloqueantes. Esto significa que la operación MPI_Recv no puede iniciar si la operación MPI_Send no ha iniciado. La ejecución de la operación MPI_Recv es bloqueada hasta que el buffer de recepción contenga los datos del emisor. La operación MPI_Send no puede iniciar si la correspondiente operación de MPI_Recv no ha iniciado. (Fig. 2.7).

  La operación MPI_Send es bloqueada hasta que el buffer de envío pueda

ser usado nuevamente. También existen operaciones no bloqueantes como MPI_Isend, MPI_Irecv. El comportamiento exacto depende de la biblioteca de MPI usada. Los siguientes dos comportamientos pueden ser observados [21]:

- Si el mensaje es enviado desde el buffer de envio sin usar un buffer del sistema interno, entonces la operacion MPI_Send es bloqueada hasta que llegue un mensaje indicando que el receptor está listo. Esto requiere que el receptor haya iniciado la operacion MPI_Recv.
- Si el mensaje es copiado a un buffer del sistema interno en tiempo de ejecución, el emisor puede seguir operando tan pronto como el copiado de los datos haya finalizado. Por lo tanto, la operacion MPI_Recv correspondiente no necesariamente tuvo que ser inicializada. Esta forma de operar tiene la ventaja de que el emisor no esta bloqueado por mucho tiempo pero tiene la desventaja que el buffer del sistema requiere espacio adicional en memoria y el copiado hacia el buffer requiere tiempo adicional de ejecución.

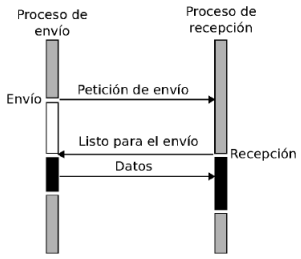

Figura 2.7: Comunicacion punto a punto.

- Comunicación colectiva [21]:

  La comunicación colectiva es un conjunto de funciones que ayudan a distribuir los datos entre los procesos, una función colectiva es ejecutada por todos los procesos en el grupo con los argumentos correspondientes. Uno de los argumentos clave es un intra comunicador que define el grupo de procesos participantes y provee un dominio de comunicación para la operación. En la ejecución de una función colectiva un proceso es definido como Maestro, donde algunos argumentos son importantes solo para el Maestro y son ignorados por los otros procesos excepto el Maestro y viceversa. Una de las actividades que realiza el proceso Maestro es transmitir un bloque de datos a todos los procesos o dividir un bloque

en partes iguales y transmitir cada parte a los procesos, dependiendo de la función que se use. También existen funciones colectivas donde varia la cantidad de datos a transmitir a cada proceso. MPI soporta una gran variedad de funciones colectivas. Las operaciones básicas soportadas en algunos lenguajes de programación son: Broadcast, Scatter, Gather. En un broadcast, un proceso envía el mismo mensaje a cada miembro del grupo. Una operación scatter permite que un proceso envié un mensaje diferente a cada miembro. Una operación gather, establece que un proceso reciba un mensaje de cada miembro del grupo. Las operaciones básicas pueden ser combinadas para formar funciones mas complejas.

- Comunicación Síncrona y Asíncrona [21]:

  Los procesos pueden comunicarse entre sí a través de compartir espacios de memoria, ya sean variables compartidas o buffers, o a través de las herramientas provistas por las rutinas de IPC. La IPC provee un mecanismo que permite a los procesos comunicarse y sincronizarse entre sí, normalmente a través de un sistema de bajo nivel de paso de mensajes que ofrece la red subyacente. En la comunicación síncrona quien envía permanece bloqueado esperando a que llegue una respuesta del receptor antes de realizar cualquier otro ejercicio. Este tipo de comunicación se presenta en el código de programación paralela del tipo Maestro - Esclavo y en de programación paralela del tipo multipoblacional. En la comunicación asincrona quién envía continúa con su ejecución inmediatamente después de enviar el mensaje al receptor. Este tipo de comunicación puede ser aplicada también a la programación paralela del tipo multipoblacional, el cuál no se realiza en el presente proyecto debido a las limitaciones del lenguaje de programación Octave, que aún no ha implementado librerías para este tipo de comunicación.

2.1.5. PelicanHPC

El PelicanHPC [8] es una distribución de GNU Linux que se ejecuta como un *live CD* (también se puede poner en un dispositivo USB, se puede arrancar desde una partición del disco duro, o puede ser utilizado como un sistema operativo virtualizado). Si el archivo de imagen ISO se quema en un CD, el CD resultante puede usarse para arrancar la computadora. El equipo en el que se inicia PelicanHPC se conoce como el "nodo de interfaz". Es el equipo con el que interactúa el usuario. Una vez PelicanHPC se está ejecutando, un guión "pelican setup" puede funcionar. Este script configura el nodo *frontend* como un servidor de arranque en red. Después de que esto se haya hecho, otros equipos pueden tener las copias de arranque de PelicanHPC a través de la

red. Estos otros equipos se conocen como “nodos de cómputo”. PelicanHPC configura el grupo formado por el nodo *frontend* y los nodos de cómputo, para que basados en la computación paralela MPI se puede hacer [8].

2.1.6. Métricas para el desempeño de un sistema

Speedup

Fundamentalmente el speedup es una técnica de la que se obtiene una relación de mejora entre el tiempo de ejecución de un programa secuencial (con un solo procesador) y en paralelo (con p procesadores). Viene definido por la siguiente formula (Ec. 2.1) [20]:

$$S = \frac{ts}{tp} \tag{2.1}$$

donde:

ts = Tiempo de ejecucíon secuencial (en un solo procesador).

tp = Tiempo de ejecucíon en p procesadores.

A partir de esta fórmula se puede pensar aplicando una lógica sencilla que el speedup simplemente corresponderá con el número de procesadores del algoritmo paralelo. Esta afirmación se conoce como speedup ideal y está considerada como una buenísima escalabilidad del algoritmo. Sin embargo, en la práctica es bastante complicado ya que conducirá a una ganancia de velocidad lineal, donde doblando el número de procesadores se doblará la velocidad de cómputo. (Fig. 2.8)

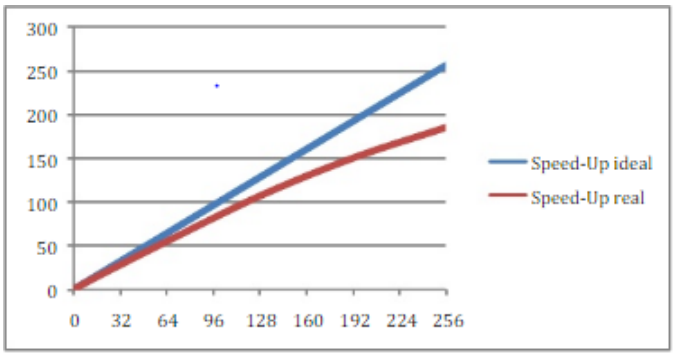

Figura 2.8: Speedup ideal y Speedup real.

Excepcionalmente puede darse un speedup superlineal, que consiste teóricamente en superar una ganancia de velocidad lineal del programa paralelo respecto al secuencial. Este fenómeno se suele dar a causa de una mejor gestión de memoria y por la acumulación de memoria cache de los procesadores.

Eficiencia

La eficiencia [20] se define en un algoritmo paralelo como el cociente entre el Speedup y el número P de procesadores. Nos proporciona una idea de la porción de tiempo que los procesadores se dedican a trabajo útil, también entendido como rendimiento (Ec. 2.2).

$$E = \frac{S}{p} \tag{2.2}$$

donde:

S = Speedup paralelo.

p = Cantidad de procesadores.

El valor máximo que puede alcanzar es uno, que corresponde con el índice de máximo aprovechamiento, ciento por ciento. Normalmente, al aumentar el número de procesadores el índice de eficiencia se aleja del valor máximo de aprovechamiento, en cambio, para un número de procesadores determinado, aumenta conforme aumente el tamaño del problema.

2.2. Algoritmos Genéticos

Los AGs son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización. Están inspirados en el proceso genético de los organismos vivos [25]. A lo largo de las generaciones, las poblaciones evolucionan en la naturaleza de acorde con los principios de la selección natural y la supervivencia de los más fuertes, postulados por Darwin. Por imitación de este proceso, los AGs son capaces de ir creando soluciones para problemas del mundo real. La evolución de dichas soluciones hacia valores óptimos del problema depende en buena medida de una adecuada codificación de las mismas. Un AG consiste en una función matemática o una rutina de software que toma como entradas a los ejemplares y retorna como salidas cuáles de ellos deben generar descendencia para la nueva generación. Versiones más complejas de AGs generan un ciclo iterativo que directamente toma a la especie (el total de los ejemplares) y crea una nueva generación que reemplaza a la antigua una cantidad de veces determinada por su propio diseño. Una de sus características principales es la de ir perfeccionando su propia heurística en el proceso de ejecución, por lo que no requiere largos períodos de entrenamiento especializado por parte del ser humano, principal defecto de otros métodos para solucionar problemas, como los Sistemas Expertos [4].

2.2.1. Lenguajes adecuados para la implementación de AGs

Los algoritmos genéticos pueden ser desarrollados en cualquier lenguaje de programación ya que los AG son métodos sistemáticos para la resolución (búsqueda, maximización o minimización) de un problema, a continuación, se mencionan algunos lenguajes que son usados para hacer aplicaciones AG: PERL, Java, Pascal, C, C++, Matlab, Tcl/Tk, PHP, Lisp, C, COBOL, Fortran, y muchos más [5]. Para este proyecto se utiliza el lenguaje Octave.

- Octave (o GNU Octave): Es un programa libre para realizar cálculos numéricos. Como su nombre indica, es parte del proyecto GNU. Es considerado el equivalente libre de MATLAB. Entre varias características que comparten, se puede destacar que ambos ofrecen un intérprete, permitiendo ejecutar órdenes en modo interactivo [14].

2.3. Antecedentes

En los últimos años, la comunidad científica internacional ha mostrado un creciente interés en una nueva técnica de búsqueda basada en la teoría de la evolución y que se conoce como el algoritmo genético. Esta técnica se basa en los mecanismos de selección que utiliza la naturaleza, de acuerdo a los cuales los individuos más aptos de una población son los que sobreviven, al adaptarse más fácilmente a los cambios que se producen en su entorno. Hoy en día se sabe que estos cambios se efectúan en los genes de un individuo (unidad básica de codificación de cada uno de los atributos de un ser vivo), y que sus atributos más deseables (i.e., los que le permiten adaptarse mejor a su entorno) se transmiten a sus descendientes cuando éste se reproduce sexualmente [4].

Los orígenes del verdadero paralelismo (MIMD) se remontan a Federico Luigi, Menabrea Conte y su Bosquejo de la máquina analítica inventada por Charles Babbage. IBM introdujo el IBM 704 en 1954, a través de un proyecto en el que Gene Amdahl fue uno de los principales arquitectos. Se convirtió en el primer equipo disponible en el mercado que utilizaba comandos aritméticos de punto flotante totalmente automáticos. Las computadoras paralelas SIMD se remontan a la década de 1970. La motivación detrás de las primeras computadoras SIMD era amortizar el retardo de la compuerta de la unidad de control del procesador en múltiples instrucciones. En 1964, Slotnick había propuesto la construcción de un ordenador masivamente paralelo para el Laboratorio Nacional Lawrence Livermore.48 Su diseño fue financiado por la Fuerza Aérea de los Estados Unidos, que fue el primer esfuerzo por lograr

la computación en paralelo SIMD. La clave de su diseño fue un paralelismo bastante alto, con hasta 256 procesadores, lo que permitió que la máquina trabajara en grandes conjuntos de datos en lo que más tarde sería conocido como el procesamiento de vectores. La computación paralela se inició como una idea debido a ciertas limitaciones de hardware. En lugar de supercomputadores para algoritmos que requerían gran poder de computo, un conjunto de CPUs de menores prestaciones podían ser utilizados para lograr las mismas o inclusive mayores prestaciones, parecía imposible su aplicación por ser un tema bastante amplio y complejo, había una enorme brecha antes de lograr ser un buen programador paralelo, lo que significa que los jefes de equipo y arquitectos informáticos se encontraban en el mismo nivel que los programadores principiantes. Luego de muchas pruebas e implementaciones, la computación paralela se ha convertido en la corriente principal cuando hablamos de problemas computacionales de gran porte.

Al diseñarse MPI, se tomaron en cuenta las características más atractivas de los sistemas existentes para el paso de mensajes, en vez de seleccionar uno solo de ellos y adoptarlo como el estándar, resultando así, en una fuerte influencia para MPI los trabajos hechos por IBM, INTEL, NX, Express, nCUBE's Vernex, p4 y PARMACS. Otras contribuciones importantes provienen de Zipcode, Chimp, PVM, Chameleon y PICL. El esfuerzo para estandarizar MPI involucró a cerca de 60 personas de 40 organizaciones diferentes principalmente de EE.UU. y Europa. La mayoría de los vendedores de computadoras concurrentes estaban involucrados con MPI, así como con investigadores de diferentes universidades, laboratorios del gobierno e industrias. El proceso de estandarización comenzó en el taller de estándares para el paso de mensajes en un ambiente con memoria distribuida, patrocinado por el Centro de Investigación en Computación Paralela en Williamsburg, Virginia, Estados Unidos (abril 29-30 de 1992). Se llegó a una propuesta preliminar conocida como MPI1, enfocada principalmente en comunicaciones punto a punto sin incluir rutinas para comunicación colectiva y no presentaba tareas seguras. El estándar final por el MPI fue presentado en la conferencia de Super cómputo en noviembre de 1993, constituyéndose así el foro para el MPI.

El proyecto PelicanHPC es una evolución de ParallelKnoppix, el cual fuera un Knoppix remasterizado para clustering. Un proyecto que fue desarrollado por Michael Creel para su propio trabajo de investigación sobre clustering, con un sencillo agregado de paquetes. A todo esto, le agregó PVM, herramientas cluster del tipo ganglia monitor, aplicaciones como GROMACS, y muchas más. También le incluyó algunos ejemplos simples de la informática paralela en Fortran, C, Python, y Octave para proporcionar algunos ejemplos básicos de trabajo para principiantes. La versión del PelicanHPC 2.2 fue anunciada el 21 de julio del 2010 disponible únicamente para amd64. El

Proyecto agradece a Robert G. Petry, Ahora es posible configurar la interfaz para arrancar sin necesidad de intervención alguna, y así enviar paquetes Wake-on-LAN a los nodos. El 19 de junio de 2013 fue el fin de la vida de PHPC, no se harán más liberaciones. La página de descarga estará disponible por un tiempo. Si alguien quiere hacerse cargo de desarrollo, la escritura de la estructura make_pelican tiene que adaptarse a las nuevas versiones de *live-build*, con el fin de hacer que las imágenes basadas en Debian Wheezy. La última versión, make pelican v2.9 utiliza live-build v2. xy, y construye imágenes basadas en Squeeze. El 28 de noviembre del 2014 Aissam Hidoussi se ha hecho cargo de desarrollo de PelicanHPC, y ha hecho un comunicado de la versión 3.0 que se basa en la actual versión estable de Debian Wheezy. El 20 de febrero del 2015 Aissam Hidoussi ha lanzado la versión 3.1 en versiones de Gnome y XFCE [12].

Octave o GNU Octave , el proyecto fue creado alrededor del año 1988, pero con una finalidad diferente: ser utilizado en un curso de diseño de reactores químicos. Posteriormente, en el año 1992, se decidió extenderlo, y comenzó su desarrollo a cargo de John W. Eaton.1 La primera versión alpha fue lanzada el 4 de enero de 1993. Un año más tarde, el 17 de febrero de 1994, apareció la versión 1.0. Octave o GNU Octave ,el proyecto fue creado alrededor del año 1988, pero con una finalidad diferente: ser utilizado en un curso de diseño de reactores químicos. Posteriormente, en el año 1992, se decidió extenderlo, y comenzó su desarrollo a cargo de John W. Eaton.1 La primera versión alpha fue lanzada el 4 de enero de 1993. Un año más tarde, el 17 de febrero de 1994, apareció la versión 1.0. El nombre surge de Octave Levenspiel, profesor de uno de los autores y conocido por sus buenas aproximaciones, por medio de cálculos elementales, a problemas numéricos en ingeniería química [14]. El 23 de de marzo de 2016 se lanza Octave 4.0.1 .Octave 4.0.1 es una versión de corrección de errores. Octave 4.0, publicado en mayo de 2015 representaba una versión nueva e importante con muchas características nuevas, incluyendo una interfaz gráfica de usuario, soporte para programación orientada a objetos classdef, una mejor compatibilidad con Matlab, y muchas funciones nuevas. El 2 de julio de 2016 se lanza Octave 4.0.3 ,ha sido liberado y ahora está disponible para su descarga. Esta versión es otra versión de corrección de errores [14].

Se realizaron varios proyectos utilizando PelicanHPC que fueron publicados en el sitio oficial de PelicanHPC [27].

Algunos trabajos anteriores referentes al tema:

- Ajuste coordinado de estabilizadores de sistemas de potencia usando algoritmos genéticos. Por: Antonio Luis Bergamo, 2000 [1].
- Econometrics. Por Michael Creel, 2008 [11].

- Ajuste Coordinado de Estabilizadores de Sistemas de Potencia y Reguladores Automáticos de Tensión utilizando Algoritmos Genéticos Multi-Objetivos. Por: Enrique Chaparro - Articulo Cientifico FPUNE, 2009 [10].
- Lagrangean Relaxation Parallel Method for Optimizing of a Hydroelectric Generation System. Por: Fanny González, Luís Fariña, Eustaquio Martńez, Esteban Vargas y Anastacio Arce, 2011 [26].
- Ajuste Coordinado de Controladores de Sistemas Eléctricos de Potencia para Mejora de la Estabilidad Angular y de Tensión usando Algoritmos Genéticos. Por: Manuel L. Sosa y Enrique Chaparro, 2013 [2].

Capítulo 3

Control de Sistemas Eléctricos de Potencia

El análisis desde el punto de vista de pequeñas perturbaciones se utiliza para el ajuste de controladores de máquinas síncronas. Estos controladores se utilizan para proporcionar a las máquinas la cantidad de torque necesaria para mejorar el desempeño dinámico general del sistema. Dicha técnica de análisis lineal proporciona información sobre las características inherentes al sistema. Hay dos formas de mejorar el desempeño dinámico de un sistema de Potencia. Una forma es aumentando el torque de sincronización de las maquinas a través de Reguladores Automáticos de Tensión (RAT) y la otra es aumentando el torque de amortiguamiento a través de señales ESP. [1]

3.1. Modelo del RAT

Fue utilizada una variante del RAT derivado del modelo ST1A [16], cuyo diagrama de bloque es mostrado en la figura (Fig. 3.1). Los parámetros a ser

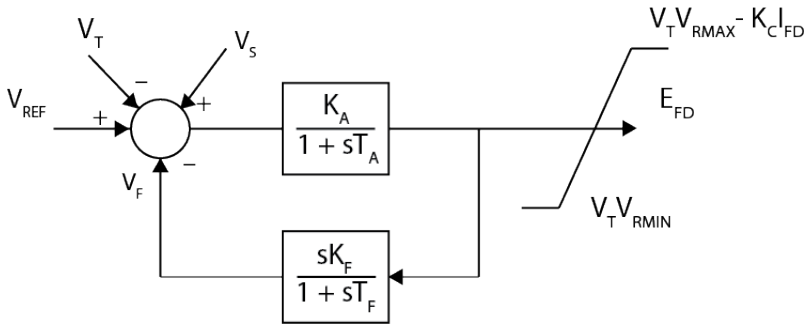

Figura 3.1: Modelo del RAT.

ajustados son: la ganancia KA y la constante de tiempo TA. Los valores de la ganancia KF son pequeños y se encuentran en el intervalo 0,02 y 0,06 pμ; y la constante de tiempo TF es normalmente mantenido constante, y cerca de 1s.

3.2. Modelo ESP

El modelo de Estabilizadores de Sistemas de Potencia (ESP) utilizado es mostrado en la figura (Fig. 3.2). La señal de entrada corresponde a la velocidad angular del rotor del generador ω, también se describen a las constantes de tiempo de los bloques de avance-atraso. El valor preciso de la constante de tiempo Tw no es crítico, y puede asumir valores comprendidos entre 3 y 10 segundos. Por lo tanto, Tw es considerado un parámetro conocido. De esa manera, los parámetros del ESP, que deberán ser ajustados por el procedimiento propuesto de ajuste coordinado son: K_s, α, y ω:

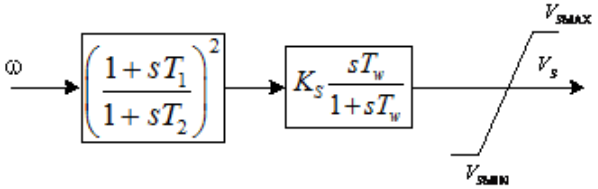

Figura 3.2: Modelo del ESP.

donde:

$$T_1 = \frac{\sqrt{\alpha_p}}{\omega_p} \tag{3.1}$$

$$T_2 = \frac{1}{\omega_p\sqrt{\alpha_p}} \tag{3.2}$$

3.3. Modelo matemático del Sistema de Potencia

El modelo del SEP, usualmente adoptado en los estudios de transitorios electromecánicos, es dado por un conjunto de ecuaciones diferenciales no-lineales de primer orden y un conjunto de ecuaciones algebraicas:

$$\frac{d_x}{d_t} = f(x,y) = f(x,V) \tag{3.3}$$

$$0 = g(x, y) = g(x, V) \tag{3.4}$$

En las ecuaciones 3.3 y 3.4, x es el vector de estados de la dinámica del SEP y V es el vector de las variables de la red eléctrica. Para estudios de estabilidad de pequeña señal, el comportamiento del SEP en torno a un punto de operación puede ser estudiado usando la forma linealizada de las expresiones 3.3 y 3.4 que representan al SEP:

$$\begin{bmatrix} \frac{d_x}{d_t} \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{J_1} & \mathbf{J_2} \\ \mathbf{J_3} & \mathbf{J_4} \end{bmatrix} \cdot \begin{bmatrix} \Delta x \\ \Delta V \end{bmatrix} \tag{3.5}$$

donde J_1 hasta J_4 son las submatrices del Jacobiano. La ecuación 3.5 representa el sistema de malla abierta del SEP; es decir, aún no fueron incluidos de los RAT y los ESP en cada generador. En la figura (Fig. 3.3) es representado el diagrama de bloque del sistema de malla cerrada del SEP:

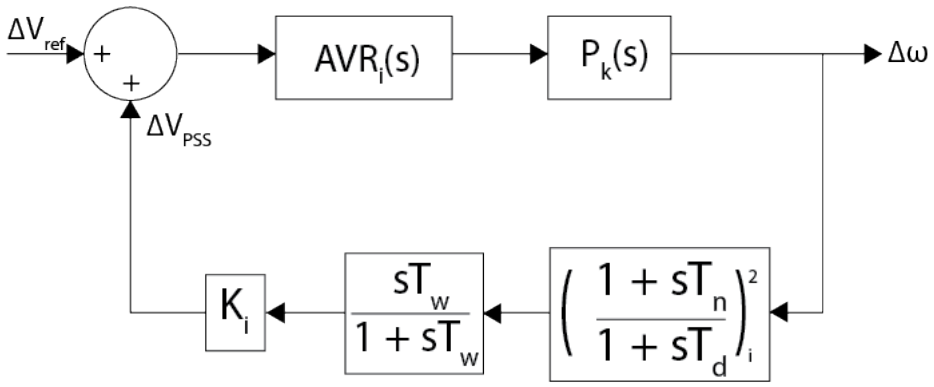

Figura 3.3: Modelo de Malla cerrada del SEP.

donde k ϵ 1, 2,.. , m representa cada condición de operación, y i ϵ 1, 2,.. , p representa cada RAT y ESP a ser optimizado por el procedimiento de ajuste.

3.4. Ajuste coordinado de los RAT y los ESP

El problema de ajuste coordinado consiste en encontrar un conjunto de parámetros de los controladores tal que optimice los criterios de desempeño del SEP bajo estudio. En el caso de los RAT y los ESP, los criterios son los siguientes [18]:

- Adecuado nivel de tensión terminal en régimen permanente de los generadores.
- Adecuado desempeño transitorio en el caso de grandes perturbaciones.

- Adecuado desempeño transitorio en el caso de pequeñas perturbaciones.

Los ESP deberán ser ajustados de tal forma que el SEP en análisis presente un adecuado grado de amortiguamiento frente a pequeñas perturbaciones. [2]

3.4.1. Ajuste del RAT

El ajuste del RAT, correspondiente a un determinado generador, debe satisfacer tres requisitos de desempeño:

- Regulación de la tensión terminal, en el generador, en régimen permanente;
- Regulación en régimen transitorio para grandes disturbios;
- Regulación en régimen transitorio para pequeñas perturbaciones;

Según la referencia (IEEE Tutorial, 1980), los tres requisitos, durante el proceso de ajuste del RAT de cada generador, constituyen un compromiso entre alta respuesta inicial y buena regulación de tensión en régimen permanente. [2]

Regulación en régimen permanente

Una de las funciones del RAT es mantener la tensión terminal del generador en un valor pre-establecido, aún cuando se produzcan variaciones en la tensión de campo del generador debido a variaciones en el punto de operación del sistema. Para el RAT mostrado en la figura (Fig. 3.4) , un alto valor en la ganancia del amplificador, juntamente con la presencia del bloque de realimentación, consiguen una rápida variación en la corrección de los desvíos transitorios de la tensión. Si K_A es expresada en p.u., desconsiderando los efectos de saturación de la máquina, el error de tensión de régimen permanente es aproximadamente $\frac{1}{K_A}$. De esta manera, para obtener una buena regulación de tensión se sugiere estimar el valor de la ganancia del regulador haciendo $K_A = \frac{T'_{dO}}{2.T_A}$, lo que resulta en valores que se encuentran en el rango de 200 y 400 p.u. (DEMELLO y CONCORDIA, 1969).

Desempeño frente a grandes disturbios

Pequeños disturbios suceden continuamente en un sistema debido a pequeñas variaciones en la carga y en la generación; y por lo tanto, poseen un comportamiento lineal, lo que permite linealizar las ecuaciones matemáticas

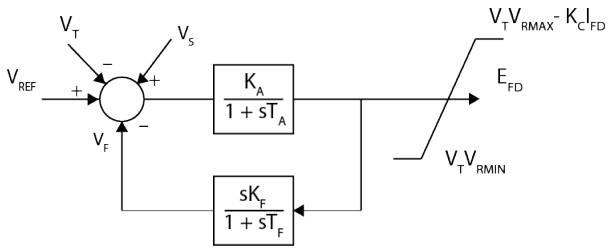

Figura 3.4: Regulador de Tensión ST1A con Realimentación.

que modelan el comportamiento dinámico del sistema en un punto de operación. De esta manera, el desempeño del sistema de excitación para pequeñas perturbaciones puede ser analizado a través de respuesta transitoria (simulación en el tiempo), respuesta en frecuencia (en términos de ganancia y fase en el diagrama de Bode) y Análisis Modal (a través de cálculo de autovalores).

La Figura(Fig. 3.5) muestra una respuesta transitoria típica de un sistema con realimentación, sometido a una pequeña variación en escalón en la señal de entrada (IEEE Guide, 1990): Los índices que miden la calidad del ajuste de los parámetros del RAT son: el tiempo de subida T_r, el overshoot y el tiempo de adecuación T_s. El objetivo en el ajuste de los parámetros del RAT, utilizando esta metodología, consiste en la minimización simultánea de los tres índices mencionados. Con un valor alto para K_A y un valor bajo para K_F (para modelos de RAT de primer orden con realimentación, como mostrado en la Fig. 3.4), disminuirá el valor de tiempo de subida, sin embargo, los valores para el overshoot y para el tiempo de asentamiento aumentan. Por lo tanto, el ajuste del RAT se convierte en un compromiso entre respuesta rápida y respuesta estable. Generalmente, se consideran valores aceptables de overshoot entre 5 y 15 % (IEEE Guide, 1990).

La figura (Fig. 3.6) muestra un diagrama típico de Bode de un sistema de malla abierta; esto es, cuando ninguno de los generadores posee ESP en el sistema de excitación asociado. Los principales índices que determinan las características de la respuesta en frecuencia de la (Fig. 3.6), para un determinado ajuste de RAT, son los siguientes:

- Ganancia de Frecuencia Baja G. Valores grandes de este índice proveen una mejor regulación de la tensión en régimen permanente;
- Frecuencia de Cruzamiento de Ganancia ω_c. Valores mayores indican respuestas más rápidas;
- Margen de Fase ϕ_m. Valores mayores proveen más estabilidad (mayor amortiguamiento post-disturbio);

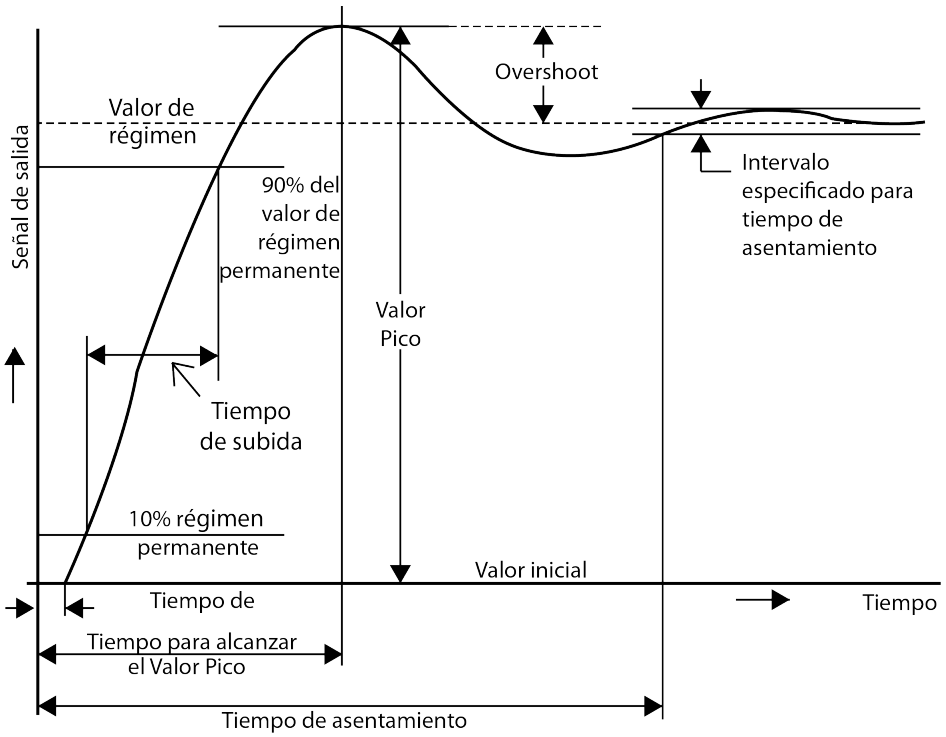

Figura 3.5: Respuesta transitoria típica de un sistema con realimentación.

- Margen de Ganancia G_m. Valores mayores proveen más estabilidad (mayor amortiguamiento post-disturbio).

En un procedimiento de ajuste, cuando es mejorado el valor de uno de los índices, los valores de otros índices son empeorados. Esto sucede también cuando se utiliza la metodología de la respuesta en frecuencia, descrita en la (Fig. 3.6), en la cual, por ejemplo, un aumento en el valor de la ganancia K_A, moverá hacia arriba tanto la curva de Bode para la ganancia (dado en decibeles, dB) como la curva asociada a la fase, lo que aumentará el valor de G y de ω_c, pero disminuirá el valor de ϕ_m y de G_m. Por lo tanto, un margen de fase de 40°C y un margen de ganancia igual a 6 dB son considerados buenos para obtener un RAT que produzca una salida estable (amortiguada) y no oscilatoria (CHAPARRO, 2007).

La figura (Fig. 3.7) muestra una respuesta en frecuencia típica de un sistema de malla cerrada. En este caso, la malla de realimentación está conectada. Por lo tanto, la entrada de la señal en el RAT corresponde al error de comparación entre la tensión de referencia y la tensión terminal del generador. Los índices de interés para el ajuste y análisis de estabilidad son:

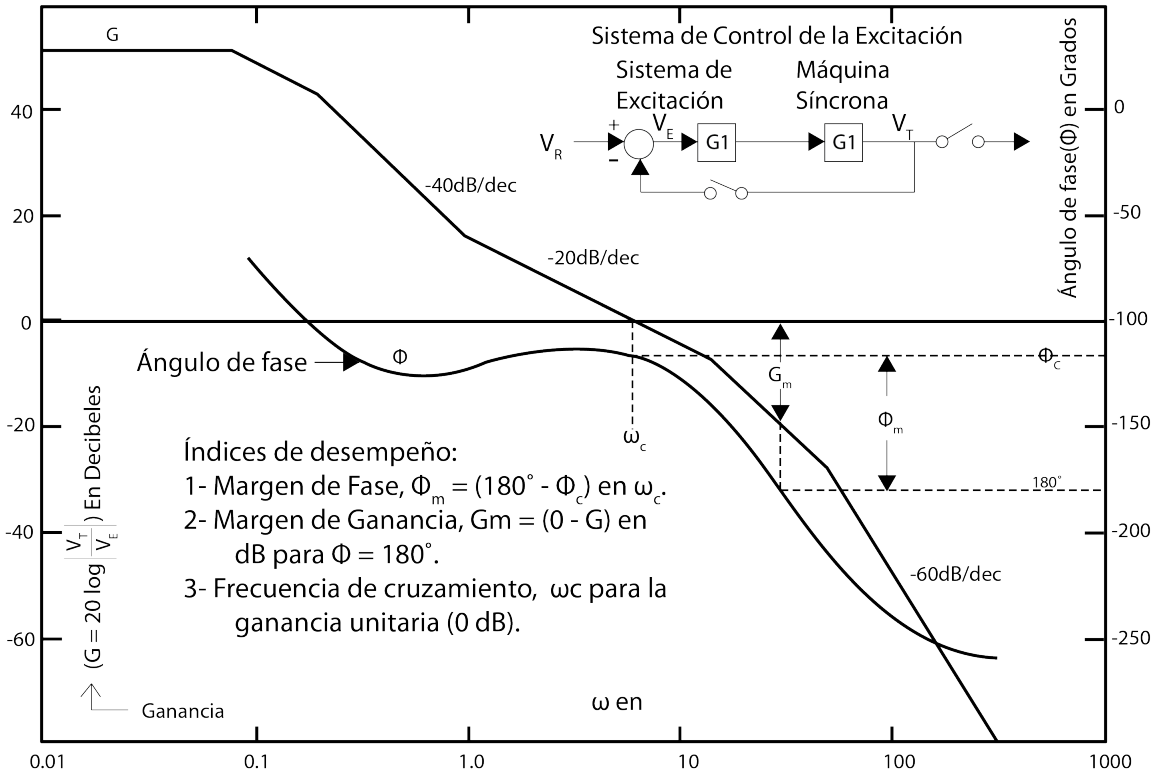

Figura 3.6: Diagrama de Bode típico de un sistema de malla abierta.

- Ancho de Banda ω_b. Valores mayores indican respuestas más rápidas;
- Valor de Pico M_p. Valores mayores indican una respuesta más oscilatoria. Un valor de M_p entre 1,1 y 1,5 dB es considerado como un buen compromiso entre respuesta rápida y oscilatoria (IEEE Tutorial, 1980). [2]

3.4.2. Ajuste del ESP

El ajuste de los SEP está basado en la linealización del modelo de SEP descripto anteriormente. El objetivo es obtener una solución en el cual los coeficientes de amortiguamiento del SEP de malla cerrada sean suficientemente altos que asegure un amortiguamiento rápido de las oscilaciones electromecánicas. [19]

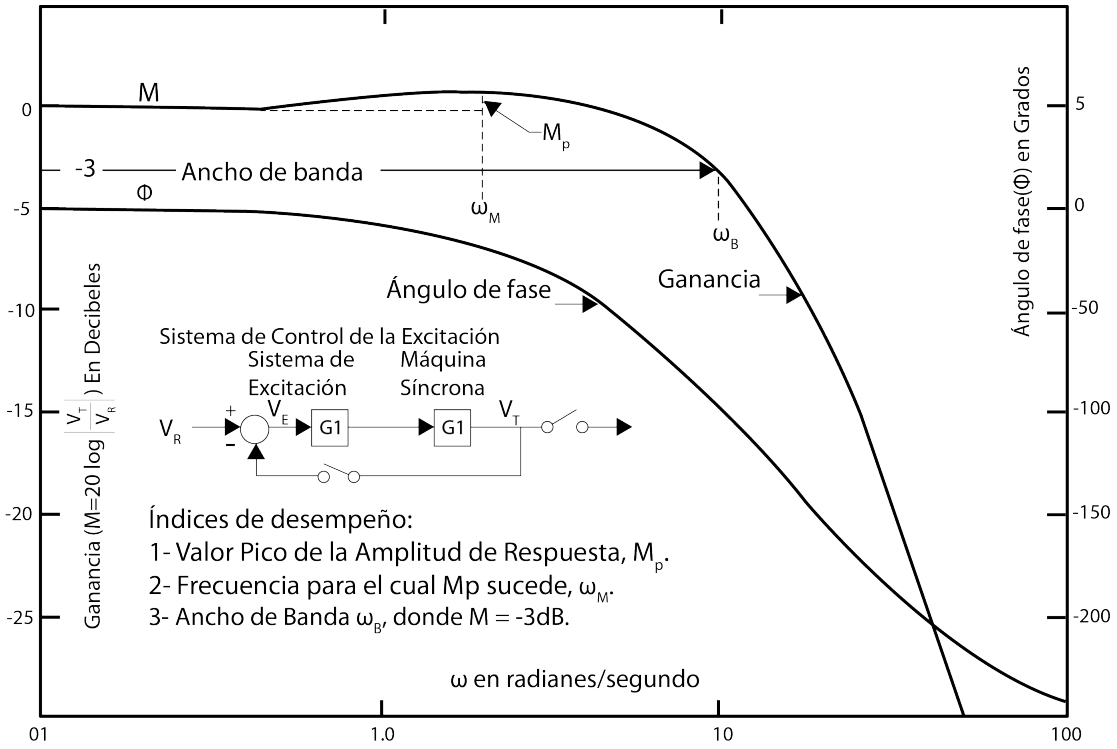

Figura 3.7: Diagrama de Bode típico de un sistema de malla cerrada.

3.5. Sistema Eléctrico de Potencia de gran porte

Un sistema de gran porte tiene como características que es intrínsecamente multimodal y requiere de un ajuste muy complejo. La aplicación de una herramienta con características robustas para lograr un ajuste coordinado es muy importante. Estos sistemas, por la gran extensión geográfica que ocupan, por los niveles de tensión en que funcionan y por la gran cantidad de energía eléctrica que transporta, requieren de la supervisión y del comando a distancia, lo cual se realiza en los Centros de Operación y Control a través de los Sistemas scada.

Debido a que el funcionamiento de los sistemas eléctricos de alternado tiene un comportamiento dinámico, las condiciones de funcionamiento deben ser establecidas aplicando criterios de funcionamiento muy estrictos para evitar los problemas de estabilidad dinámica, que pueden llevar al sistema al estado de colapso. Se considera de gran porte a un sistema con más de 50 subestaciones y más de 500 generadores.

Capítulo 4

Metodología para la implementación de Computación Paralela a Algoritmos Genéticos

El modelo matemático de SEP, adoptado en estudios de transitorios electromecánicos, es representado por un conjunto de ecuaciones diferenciales no lineales de primer orden, que describen el desempeño dinámico de los generadores síncronos y de los compensadores estáticos instalados, y por un conjunto de ecuaciones algebraicas, como expresa la ecuación 4.1 Las ecuaciones algebraicas determinan el estado de la red eléctrica en régimen permanente y las relaciones de potencia entre los generadores y las correspondiente barras terminales, asociadas a las subestaciones de elevación de tensión para la transmisión (KUNDUR, 1993). Para los estudios de estabilidad frente a pequeñas perturbaciones, el desempeño del SEP, alrededor de un determinado punto de operación, puede ser linealizado, resultando en una expresión de espacio de estados descrita en la siguiente ecuación matricial:

$$I(x, V) = Y_N.V \tag{4.1}$$

$$\begin{bmatrix} \frac{d_x}{d_t} \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{J_1} & \mathbf{J_2} \\ \mathbf{J_3} & \mathbf{J_4} \end{bmatrix} . \begin{bmatrix} \Delta x \\ \Delta V \end{bmatrix} \tag{4.2}$$

La matriz de la ecuación 4.2 se denomina Jacobiano Expandido. Cada una de las componentes de la submatriz J_1 contiene a los coeficientes asociados a todas las variables de estado correspondientes a los generadores síncronos y a sus controladores del sistema de excitación, así como aquéllos asociados a los compensadores estáticos del SEP. El resto de las submatrices, J_2 y J_3,

relacionan el desempeño de los generadores y de los compensadores estáticos con la red; y, las componentes de la submatriz J_4 provienen de las ecuaciones algebraicas que describen el desempeño de la red. Las expresiones matemáticas de espacio de estados que resultan de los modelos del RAT, ESP y del CER utilizados en este trabajo, respectivamente, son las siguientes:

$$E_{FD} = \frac{k_A}{T_A}.V_S + \frac{k_A}{T_A}.V_{REF} - (\frac{k_A}{T_A}.\frac{k_F}{T_F}.\frac{1}{T_A}).E_{FD} - \frac{k_A}{T_A}.V_{REF} - \frac{k_A}{T_A}.V_{TI} \quad (4.3)$$

$$X_1 = -\frac{1}{T_2}.X_1 + (\frac{T_2 - T_1}{T_2^2}).\omega \quad X_2 = -\frac{1}{T_2}.X_2 + (\frac{T_2 - T_1}{T_2^2}).X_1 \quad X_3 = K_s.X_2 \quad (4.4)$$

$$B = -\frac{1}{T_{CER}}.B + \frac{K_{CER}}{T_{CER}}.V_C \quad (4.5)$$

Los coeficientes de las ecuaciones de estado 4.3, 4.4 y 4.5, correspondientes a los modelos del RAT, del ESP y del CER utilizados, se encuentran en la submatriz J_1, y serán modificados por la metodología de ajuste coordinado propuesta para encontrar valores óptimos, asociados a los referidos controladores. En la ecuación 4.3, V_{TR}y V_{TI} representan las componentes, real e imaginaria, respectivamente, de la tensión terminal del generador, donde se encuentra instalado el RAT.

4.1. Descripción de la Metodología de Ajuste Coordinado

El problema de ajuste coordinado consiste en encontrar un conjunto de parámetros de los controladores del sistema de excitación de cada generador y los correspondientes parámetros de los grupos de CER, de manera que éstos perfeccionen los criterios de desempeño del SEP en estudio. En el caso de los RAT y de los ESP, estos criterios son los siguientes (KUNDUR, 1993):

- Adecuado nivel de tensión terminal de los generadores en régimen permanente;
- Adecuado desempeño transitorio en el caso de grandes perturbaciones;
- Adecuado comportamiento transitorio en el caso de pequeñas perturbaciones.

Sin embargo, para los compensadores estáticos, los criterios de desempeño esperados son:

- Compensación de reactivos en el caso de grandes disturbios;
- Adecuado perfil de tensión para todas las subestaciones del sistema multi-máquina.

Por lo tanto, para que los controladores y compensadores descritos satisfagan estos criterios, sus parámetros fueron ajustados a través de una metodología de ajuste coordinado, basado en los AG, que considera un conjunto de condiciones de operación pre-establecidas. La ecuación matricial 4.2 será obtenida de la linealización de cada condición de operación pre-establecida. Luego, todos los parámetros de los controladores del sistema de excitación de cada generador, así como aquéllos correspondientes a los compensadores estáticos, calculados a través de la metodología de ajuste coordinado propuesta, serán considerados en cada condición de operación para la obtención de la matriz de malla cerrada a través de la reducción gaussiana (KUNDUR, 1993). Utilizando la técnica de Análisis Modal, serán calculados todos los autovalores de la matriz de malla cerrada, asociada a cada condición de operación. Luego, estos autovalores serán utilizados para el cálculo del coeficiente de amortiguamiento ζ_i, tal que i $\in$ { 1, 2,. . . , mxn} | n y m, son la dimensión de la matriz de malla cerrada y el número de condiciones de operación, respectivamente. En este caso el mínimo coeficiente de amortiguamiento calculado ζ_{min}, considerando todas las condiciones de operación pre establecidas, constituye el índice de evaluación de la estabilidad del sistema.

4.2. Función Objetivo

La función objetivo considerada para el ajuste de los parámetros dinámicos del RAT, del ESP y del CER, que el AG maximizará en su proceso de búsqueda, corresponde al amortiguamiento mínimo calculado,dado por la ecuación 4.6

$$Max\{F = \zeta_{min}\} \tag{4.6}$$

4.3. Codificación y Vector Solución

Cada solución factible es representada a través de un vector, o cromosoma artificial, que el manipulará en su proceso de optimización, codificado con números reales, conforme descripto en la figura (Fig. 4.1): En la figura (Fig. 4.1), las primeras componentes corresponden a los parámetros del ESP, p es el número de estabilizadores ajustados. Las componentes del medio son las ganancias de los RAT, q es el número de reguladores ajustados. Los últimos

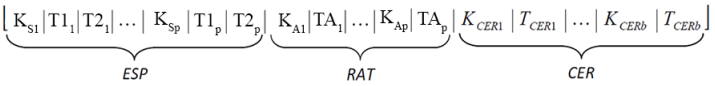

Figura 4.1: Estructura de la solución

parámetros corresponden a la ganancia y a la constante de tiempo de los reguladores automáticos de los CER, respectivamente, b es el número de barras piloto consideradas.

4.4. Población Inicial del AG

La población inicial de soluciones factibles fue creada de la siguiente manera:

- Fueron generadas aleatoriamente N−1 soluciones, satisfaciendo los límites mínimos y máximos de los parámetros asociados a cada dispositivo de control considerado. Los límites de cada parámetro son dados por la ecuación 4.7.

$$\begin{gathered} 1 \leq K_s \leq 20 p.u. \quad 100 \leq K_A \leq 400 p.u. \\ 0,02 \leq T_A \leq 0,5 s. \\ 0,05 \leq T_1 \leq 0,5 s. \quad 100 \leq K_{CER} \leq 400 p.u. \\ 0,005 \leq T_2 \leq 0,05 s. \quad 0,02 \leq T_{CER} \leq 0,5 s. \end{gathered} \tag{4.7}$$

- El individuo restante será determinado aplicando el Criterio de Nyquist para el ajuste del ESP, como se indica en (MARTINS y LIMA, 1989), en un sistema equivalente, donde cada generador es conectado a una barra infinita. En este sistema equivalente, la ganancia del RAT puede ser igual a 100 p.u., como se verá en el caso del SIS9B, o puede ser ajustado por el método tradicional de simulación dinámica, que será aplicado a los RAT del IEEE14B. Luego, considerando el SEP completo, serán estimados los parámetros del CER, de manera a obtener mínimos coeficientes de amortiguamiento positivos en cada una de las condiciones de operación críticas pre-establecidas. De esta manera, para el ajuste de los ESP, aplicando el Criterio de Nyquist, ya serán considerados los valores numéricos de los parámetros del RAT y del CER.

4.5. Operadores Probabilísticos del AG

Selección: Fue implementado el método del Torneo Estocástico en el cual, se escogen aleatoriamente un determinado número de individuos; y, de ese número de individuos, se selecciona uno, el cual correspondera a la solución cuyo Fitness sea el más alto (en un problema de maximización). En el presente trabajo, en el Torneo Estocástico se seleccionan aleatoriamente 5 (cinco) individuos. Este proceso de selección se realiza N veces, donde N es el tamaño de la población, en el AG.

Cruzamiento: Basado en la metodología de dos puntos de corte, donde los componentes que serán intercambiados, entre cada par de individuos, son aquellos del lado derecho del punto de corte, cada vez que se ejecuta el cruzamiento. El procedimiento de cruzamiento será ejecutado considerando una probabilidad pc = 0,7.

Mutación: Serán recorridos todos los individuos y todos los componentes dentro de cada individuo, en cada iteración del AG. Cuando se ejecuta la mutación, el componente afectado es intercambiado aleatoriamente por un valor diferente, respetando los límites mínimo y máximo del parámetro correspondiente. La ejecución de la mutación depende de la probabilidad de mutación pm = 0,01, que fue considerada constante durante el proceso de optimización.

4.6. Pseudocódigo del AG

El AG, adaptado para el ajuste coordinado propuesto se ilustra en la figura (Fig. 4.2): El pseudocódigo mostrado por el flujograma de la figura (Fig. 4.2) sigue los siguientes pasos para el ajuste de los parámetros de los RAT, ESP y CER de un sistema de potencia analizado:

- Generar la población inicial del AG, tal como se describió anteriormente: N−1 individuos determinados aleatoriamente; y, el individuo restante, a través del criterio de Nyquist. También se inicializa el contador de iteraciones, t = 0;

- - Modificar la matriz Jacobiana Expandida, según los valores numéricos de cada solución factible, generados por el AG, en las posiciones correspondientes a los coeficientes de las ecuaciones de estado de los modelos de RAT, ESP y CER, mostradas por las expresiones 4.3, 4.4 y 4.5, respectivamente, y encontrados en la submatriz J1, de la ecuación 4.2, para todas las condiciones de operación pre-establecidas.

- Reducir el Jacobiano Expandido, asociado a cada condición de operación, a través de la reducción gaussiana, eliminando las filas y columnas correspondientes a las variables algebraicas del SEP analizado, para obtener la matriz de estado de malla cerrada.
- Calcular todos los autovalores de la matriz de estado de malla cerrada obtenida, asociada a cada condición crítica de operación, previamente seleccionada, utilizando el método QR.
- Calcular los coeficientes de amortiguamiento a partir de los autovalores obtenidos.
- Definir el mínimo coeficiente de amortiguamiento ζ_{min}, que constituye el valor numérico de la función objetivo F, descripto en la ecuación 4.6, considerando todos los coeficientes de amortiguamiento en cada una de las condiciones de operación preestablecidas.
- Aplicar los operadores genéticos del AG a la población de soluciones factibles, para obtener nuevos vectores de solución para la siguiente iteración.
- Incrementar el contador de iteraciones t = t + 1, si el criterio de parada aún no fue satisfecho, continuando el proceso iterativo del algoritmo a partir del ítem 2, hasta que el criterio de parada sea alcanzado (Fig. 4.2). [2]

4.7. Implementación Paralela al AG

Para el algoritmo anteriormente desarrollado se implementa computación paralela del tipo síncrono, junto con una estrategia de comunicación de datos (soluciones parciales) entre los procesadores de una red de área local mediante librerías MPI, todo el proceso se desarrolla en el lenguaje de programación Octave para plataforma PelicanHPC. La plataforma PelicanHPC es iniciado en una de las computadoras y luego las demás computadoras son iniciadas a través de la red, el algoritmo es ejecutado en la primera computadora donde se inicializo el PelicanHPC, el cual actúa como Maestro, para todos los casos esta es la encargada de distribuir las tareas a cada Esclavo.

En el caso del Maestro − Esclavo, cuando cada proceso Esclavo recibe las órdenes y los datos que fueron enviados por el proceso Maestro, estos reciben, realizan los cálculos asignados y luego proceden a devolver los valores que fueron calculados, el proceso Maestro procede a recibir dichos valores y esto sucede en un ciclo iterativo. Al terminar el ciclo iterativo el proceso

Maestro muestra los resultados obtenidos (Fig. 4.3). Para el segundo caso que es el del tipo Multipoblacional, cada proceso ya sea Maestro o Esclavo contienen sus propios valores y realizan los cálculos establecidos en cada uno de ellos, al obtener el valor calculado cada proceso envía sus mejores valores a los demás procesos, cada proceso recibe esos valores y compara cual es el mejor valor de todos, esto sucede en un ciclo iterativo. Al terminar el proceso iterativo se comparan los resultados y se muestra el mejor valor obtenido (Fig. 4.4). Se verifican los resultados de ajuste obtenidos en una única computadora, del AG secuencial y se comparan con el AGP Maestro − Esclavo y Multipoblacional ejecutado en un cluster de computadoras.

4.8. Diseño

4.8.1. Paralelización Maestro - Esclavo

- Modelo del desempeño de ajuste de controladores del SEP. (Fig. 4.6)
- Inicialización de variables y comandos MPI.
- Inicialización en el procesador Maestro: inicializa semilla numérica para números aleatorios.
- En el Maestro se crea la población inicial, los valores del RAT, PSS y SVC. Los parámetros de cada valor se organizan.
- Calculo de los valores KA, TA, Ks, T1, T2, Kc y Tc.
- Realización del ajuste inicial de los controladores.
- El proceso Maestro envía la matriz xpar creada a cada uno de los Esclavos.
- Cada proceso Esclavo recibe la matriz xpar del proceso Maestro y procede a la evaluación del Fitness de la población.
- Cada proceso Esclavo envía el Fitness calculado al proceso Maestro.
- El proceso Maestro recibe los Fitness de cada proceso Esclavo y obtiene el valor máximo del Fitness.
- Inicia el ciclo iterativo.
- El proceso Maestro aplica selección, cruzamiento y mutación.

- El proceso Maestro envía la nueva matriz xpar creada a cada uno de los Esclavos.
- Cada proceso Esclavo recibe la nueva matriz xpar del proceso Maestro y procede a la evaluación de nuevos Fitness de la población.
- Cada proceso Esclavo envía el nuevo Fitness calculado al proceso Maestro.
- El proceso Maestro recibe los Fitness de cada proceso Esclavo, obtiene el valor máximo del nuevo Fitness y procede a la aplicación de Elitismo.
- Finaliza el ciclo iterativo.
- El proceso Maestro finaliza mostrando el Fitness optimo, los puntos de cruzamiento y mutación guardando los resultados en un archivo.

4.8.2. Paralelización Multipoblacional Síncrono

- Modelo del desempeño de ajuste de controladores del SEP. (Fig. 4.7)
- Inicialización de variables y comandos MPI.
- Inicialización de procesos Maestro y Esclavo paralelamente y en ambos se realizan:
- Inicialización de semilla numérica para números aleatorios.
- Creación de la población inicial, los valores del RAT, PSS y SVC. Los parámetros de cada valor se organizan tanto en el proceso Maestro como en cada proceso Esclavo.
- Calculo de los valores KA, TA, Ks, T1, T2, Kc y Tc.
- Ajuste inicial de los controladores.
- Evaluación del Fitness.
- Obtención del valor máximo de Fitness.
- Comienza ciclo iterativo en el Maestro y en los Esclavos.
- Inicialización de procesos Maestro y Esclavo paralelamente y en ambos se realizan:
- Inicialización de semilla numérica para números aleatorios.

- El proceso Maestro envía el mejor Fitness y la matriz xMpar creada a cada proceso Esclavo.
- Cada proceso Esclavo recibe el mejor Fitness y la matriz xMpar del Maestro.
- Cada proceso Esclavo envía a los demás esclavos el mejor Fitness y la matriz xEpar.
- El proceso Maestro recibe el mejor Fitness y la matriz xEpar de cada Esclavo.
- Aplicación de selección, cruzamiento y mutación.
- Evaluación de Fitness y aplicación de Elitismo.
- Finaliza el ciclo iterativo.
- El proceso Maestro finaliza mostrando el Fitness optimo, los puntos de cruzamiento y mutación guardando los resultados en un archivo.

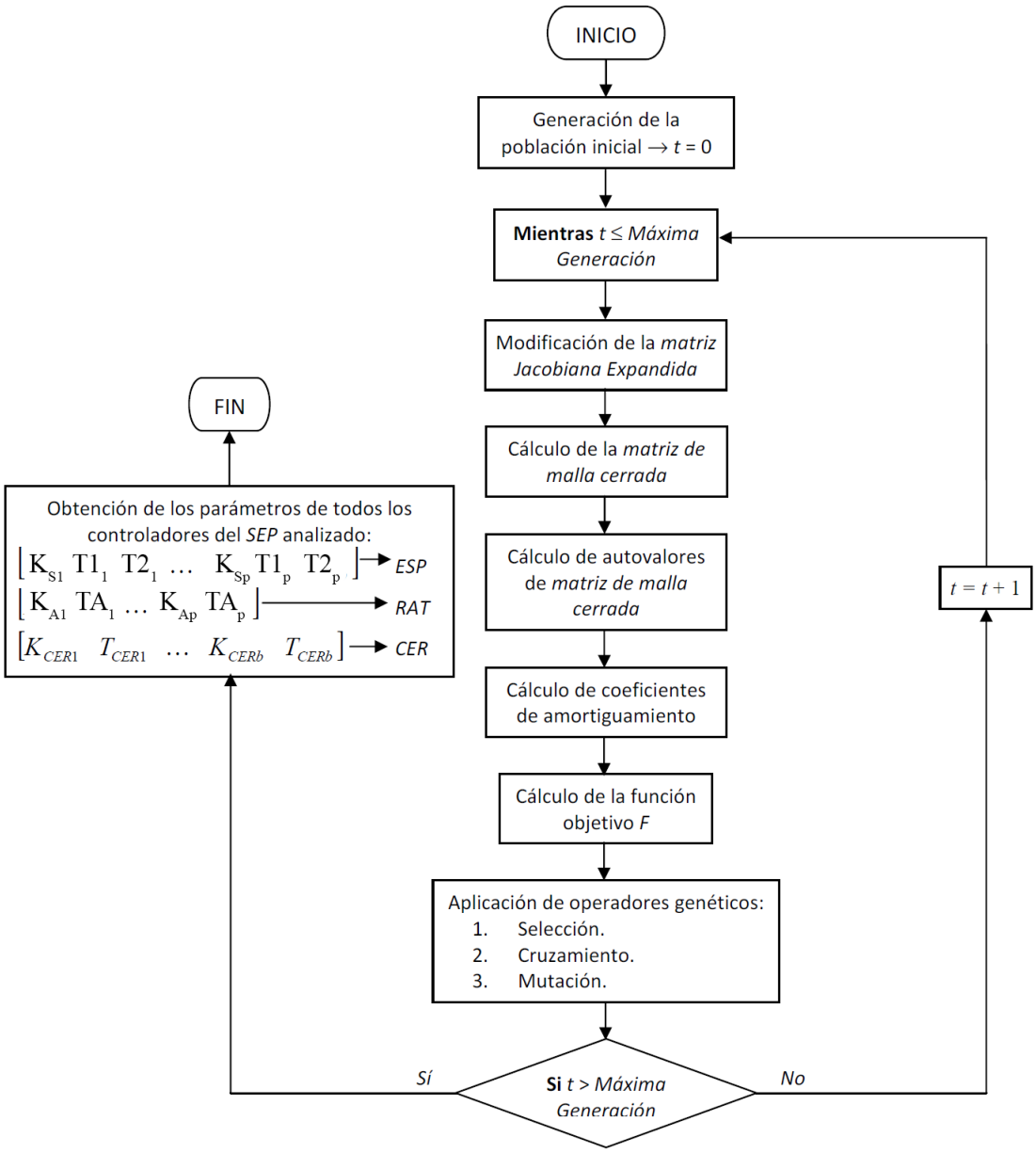

Figura 4.2: Esquema de funcionamiento del AG Secuencial.

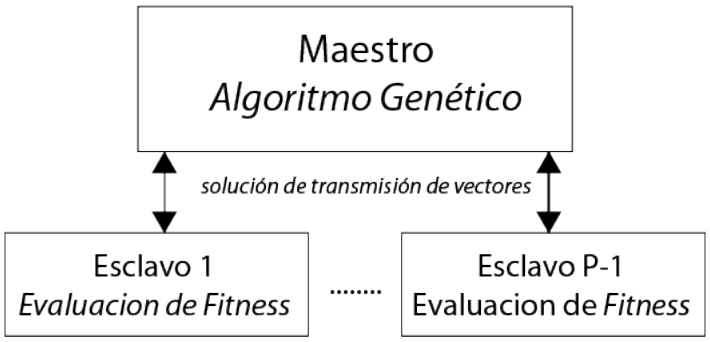

Figura 4.3: Esquema de funcionamiento del AGP Maestro−Esclavo.

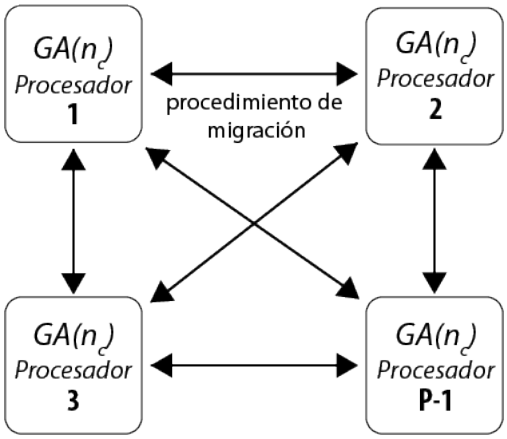

Figura 4.4: Esquema de funcionamiento del AGP Multipoblacional.

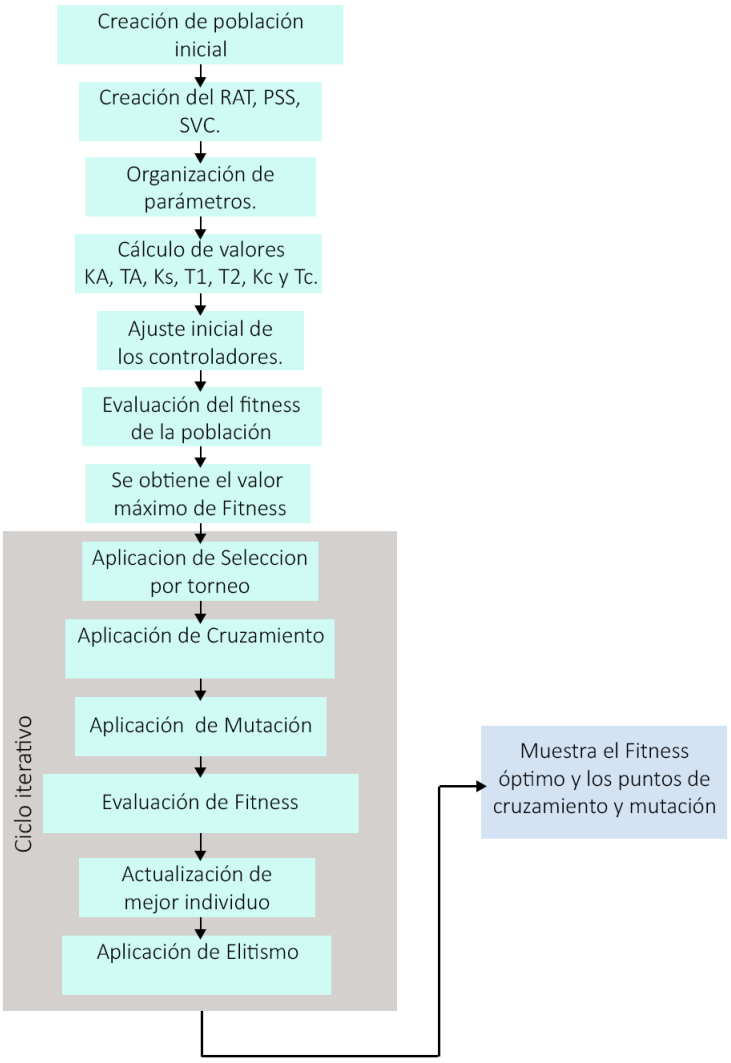

Figura 4.5: Funcionamiento del AG secuencial.

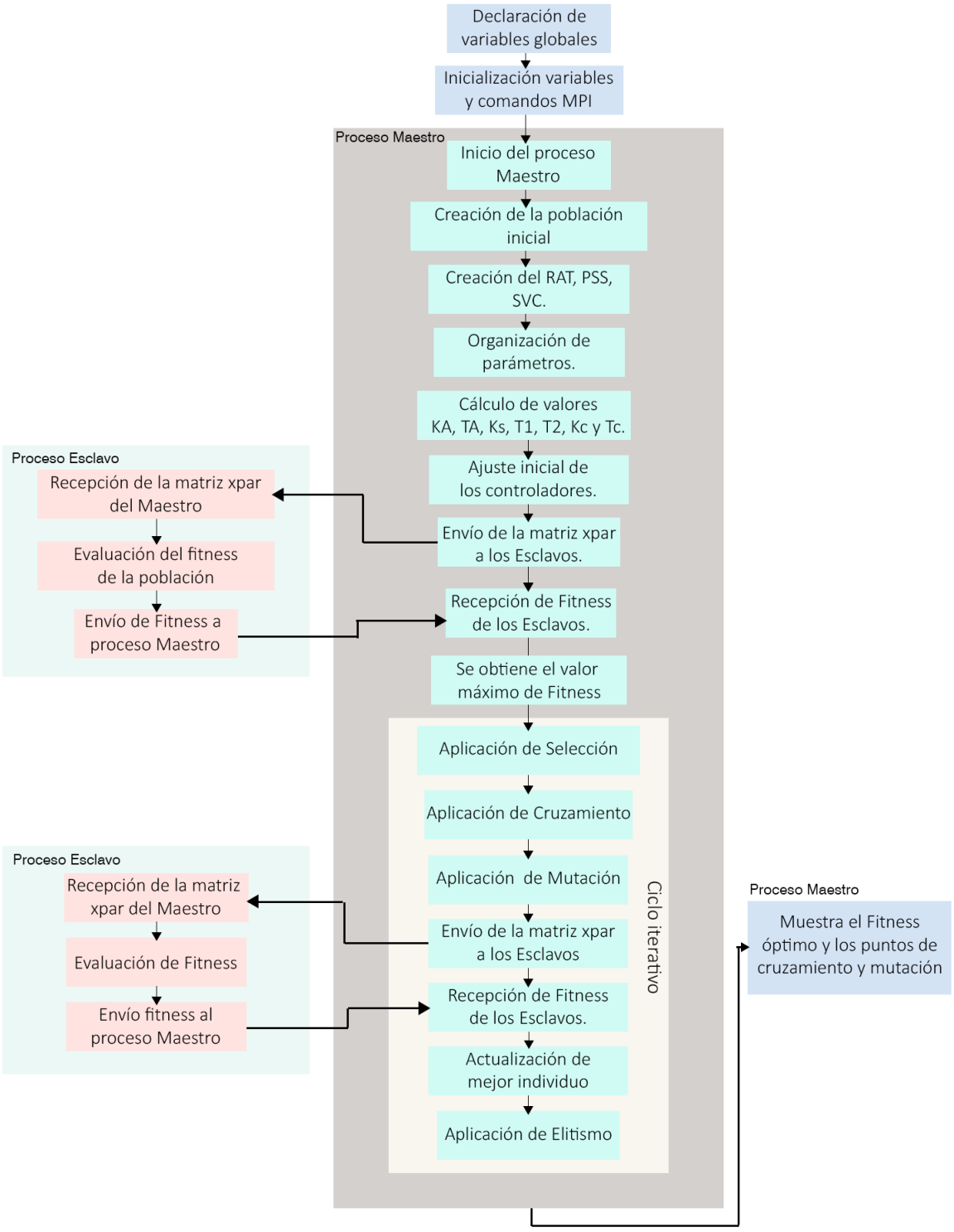

Figura 4.6: Funcionamiento del AGP Maestro - Esclavo.

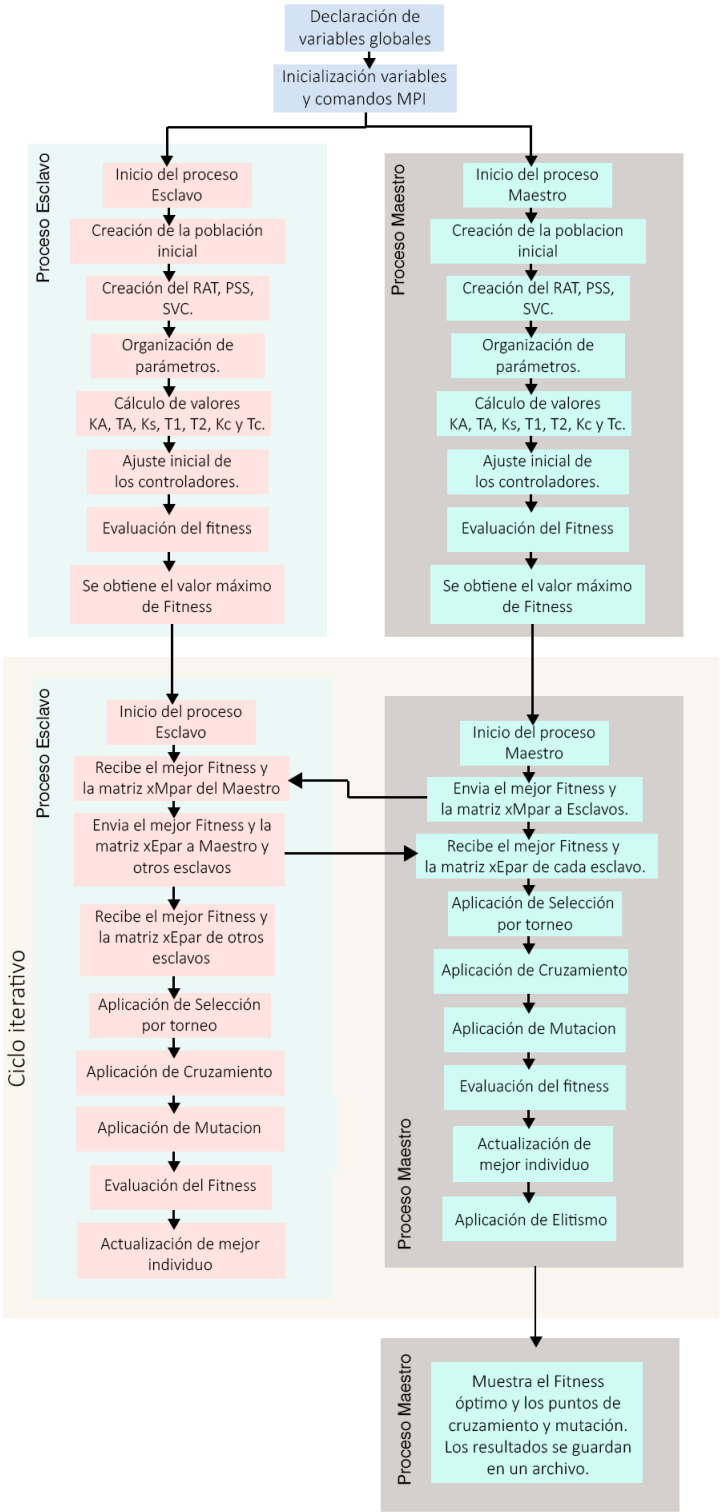

Figura 4.7: Funcionamiento del AGP multipoblacional síncrono.

Capítulo 5

Resultados

5.1. Experimentos

5.1.1. Descipción del sistema New England

El SEP utilizado para probar la metodología de ajuste coordinado propuesta fue el sistema New England, descrito en 5.1, que contiene 39 buses y 10 generadores. En este sistema hay 9 controladores de excitación (AVR y PSS) para ser ajustados simultáneamente con 3 dispositivos SVC, teniendo en cuenta seis condiciones operativas, que se describen en la Tabla 5.1. Los dispositivos SVC se encuentran en los buses 6, 7 y 20 de New England. La figura (Fig. 5.1) muestra la topología del sistema de New England.

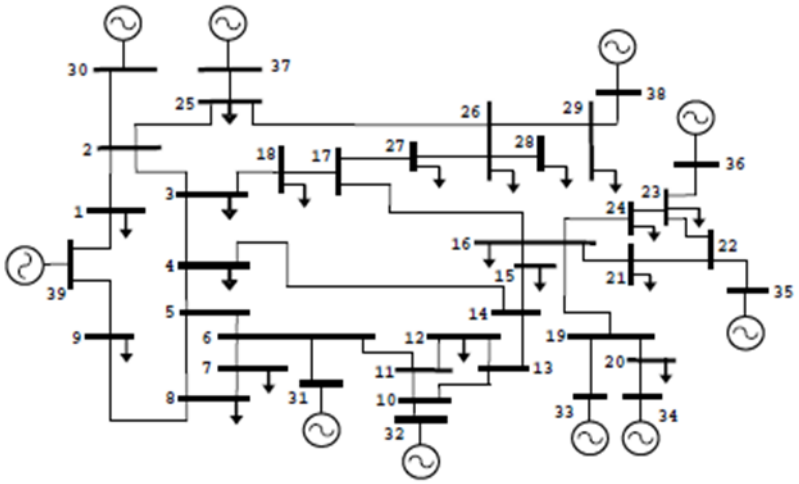

Figura 5.1: Sistema New England.

#	**Descripciones**
1	Caso Base.
2	TL * 4-14 y TL 17-17 fuera de servicio.
3	TL 3-18 y TL 25-26 fuera de servicio.
4	TL 6-11 fuera de servicio.
5	Carga total incrementada en 10 %.
6	Carga total incrementada en 10 %.

Tabla 5.1: Condiciones Operacionales.

*TL Línea de transmisión.

La experimentación se llevó a cabo en el laboratorio de informática de la Facultad Politécnica − UNE, en el cual las computadoras se encuentran conectadas en red, para las pruebas del algoritmo paralelo de ajuste coordinado de controladores del SEP del tipo Maestro − Esclavo se utilizaron siete computadoras con las siguientes características (Tabla 5.2):

Nodos	**Procesador**	**Memoria RAM**
Maestro	Intel Core 2	3.00 GB
Esclavo 1	Intel Core 2	2.00 GB
Esclavo 2	Intel Core 2	3.00 GB
Esclavo 3	Intel Core 2	2.00 GB
Esclavo 4	Intel Core 2	2.00 GB
Esclavo 5	Intel Core 2	3.00 GB
Esclavo 6	Intel Core 2	3.00 GB

Tabla 5.2: Características de las computadoras utilizadas para AGP del tipo Maestro − Esclavo.

La ejecución del algoritmo secuencial se llevó a cabo en la primera computadora mencionada en la tabla anterior, la cual actúa como Maestro en la ejecución paralela. Para todas las ejecuciones, incluyendo el secuencial, se utilizó la plataforma Pelican HPC Las distintas pruebas para el algoritmo del tipo Maestro − Esclavo varían en la cantidad de procesos que se han asignado para cada conjunto de Esclavos, de tal forma que ocurra un balanceo de carga. Para las pruebas del algoritmo paralelo de ajuste coordinado de controladores del SEP del tipo Multipoblación se utilizaron 6 (seis) computadoras con las siguientes características (Tabla 5.3):

Nodos	Procesador	Memoria RAM
Escalvo 1	Intel Core 2	3.00 GB
Esclavo 2	Intel Core 2	2.00 GB
Esclavo 3	Intel Core 2	3.00 GB
Esclavo 4	Intel Core 2	2.00 GB
Esclavo 5	Intel Core 2	2.00 GB
Esclavo 6	Intel Core 2	3.00 GB

Tabla 5.3: Características de las computadoras utilizadas para AGP Multipoblacional.

5.2. Resultados Obtenidos

5.2.1. AGP Maestro − Esclavo

Dado que se dispone de siete procesadores, las pruebas se llevaron a cabo con un (para el secuencial) dos, cuatro y seis esclavos, se procedieron a 2 ejecuciones para cada caso y los resultados obtenidos fueron los siguientes (Tabla 5.4):

Pruebas	Cant. Procesos Esclavos	Tiempo en segundos	Fitness Obtenido
Secuencial	1	2755,63	15,33
Prueba 1	2	1384,20	15,31
Prueba 2	4	708,44	15,55
Prueba 3	6	479,44	15,37

Tabla 5.4: Resultados obtenidos del AG Secuencial y del AGP Maestro − Esclavo

Promediando el tiempo los resultados se ilustran en la siguiente figura (Fig. 5.2) para una mejor apreciación:

La gran diferencia de tiempo entre la ejecución con 1 solo proceso (2755,63 segundos) y el resto de las pruebas (menor a 1410 segundos) es evidente en el gráfico presentado, a medida que incrementa la cantidad de procesos, el tiempo de ejecución disminuye casi en la mitad. Teniendo en cuenta la definición del índice de rendimiento conocido como speedup (o aceleración), se pueden graficar los valores del speedup obtenidos para cada cantidad de procesadores utilizados. Como es conocido, el valor máximo del speedup en teoría es igual a la cantidad de procesadores, razón por la cual los resultados suelen compararse con la recta y = x. Como lo muestra la figura 5.3, el

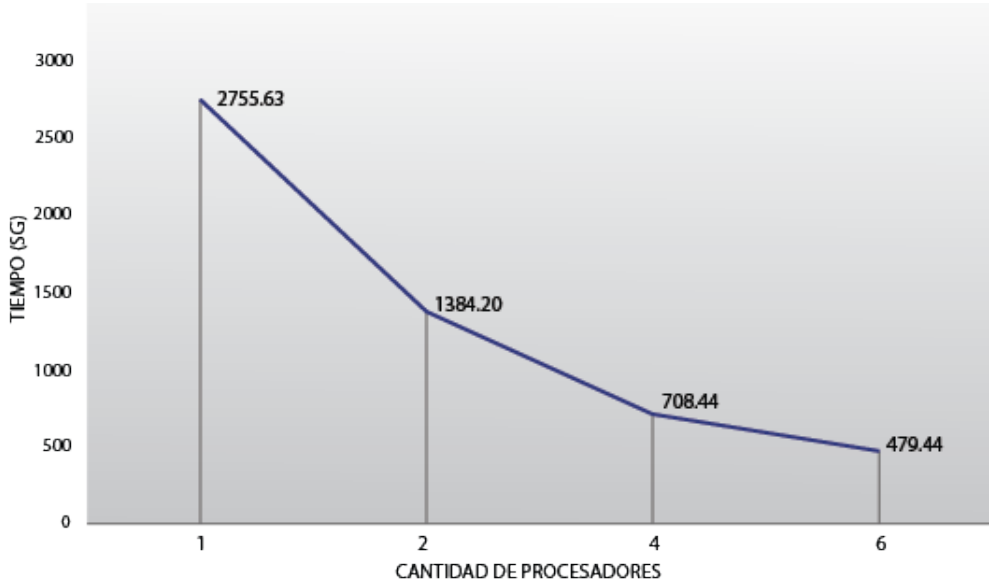

Figura 5.2: Tiempo de ejecución del algoritmo paralelo Maestro − Escalvo.

speedup obtenido es cercano al teórico, por lo tanto se considera que presenta un comportamiento sub-lineal.

Luego de haber obtenido los resultados anteriores se procedió a hallar la eficiencia y los resultados fueron los siguientes (Tabla 5.5):

Pruebas	**Cant. de Procesos**	**Speedup**	**Eficiencia (%)**
Prueba 1	2	1,99	99,54
Prueba 2	4	3,89	97,24
Prueba 3	6	5,75	95,79

Tabla 5.5: Eficiencia del AGP Maestro − Esclavo.

Dichos resultados pueden ser visualizados de forma gráfica en la siguiente figura (Fig. 5.4).

5.2.2. AGP Multipoblacional

Dado que se dispone de seis procesadores, las pruebas se llevaron a cabo con dos, cuatro y seis esclavos, se procedieron a dos ejecuciones para cada caso y los resultados obtenidos fueron los siguientes (Tabla 5.6):

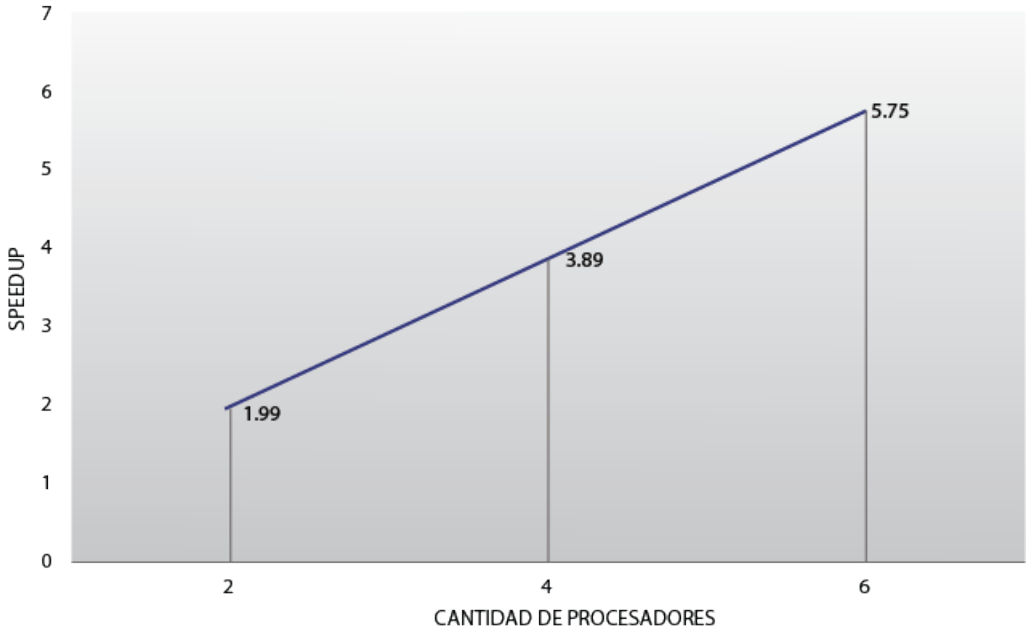

Figura 5.3: Valores de speedup del AGP Maestro − Esclavo.

Pruebas	**Cant. Procesos Esclavos**	**Tiempo en segundos**	**Fitness Obtenido**
Prueba 1	2	1400,08	15,72
Prueba 2	4	701,50	16,55
Prueba 3	6	467,88	15,68

Tabla 5.6: Resultados Obtenidos AGP Multipoblacional

Promediando el tiempo los resultados se ilustran en la siguiente figura (Fig. 5.5) para una mejor apreciación:

Teniendo en cuenta la definición del índice de rendimiento conocido como speedup (o aceleración) se pueden graficar los valores del speedup obtenidos para cada cantidad de procesadores utilizados. Como es conocido, el valor máximo del speedup en teoría es igual a la cantidad de procesadores, razón por la cual los resultados suelen compararse con la recta y = x. Como lo muestra la figura, el speedup obtenido es cercano al teórico, por lo tanto se considera que presenta un comportamiento sub-lineal (Fig. 5.6).

Luego de haber obtenido los resultados anteriores se procedió a hallar la eficiencia y los resultados fueron los siguientes (Tabla 5.7):

Dichos resultados pueden ser visualizados en forma gráfica en la siguiente figura (Fig. 5.7).

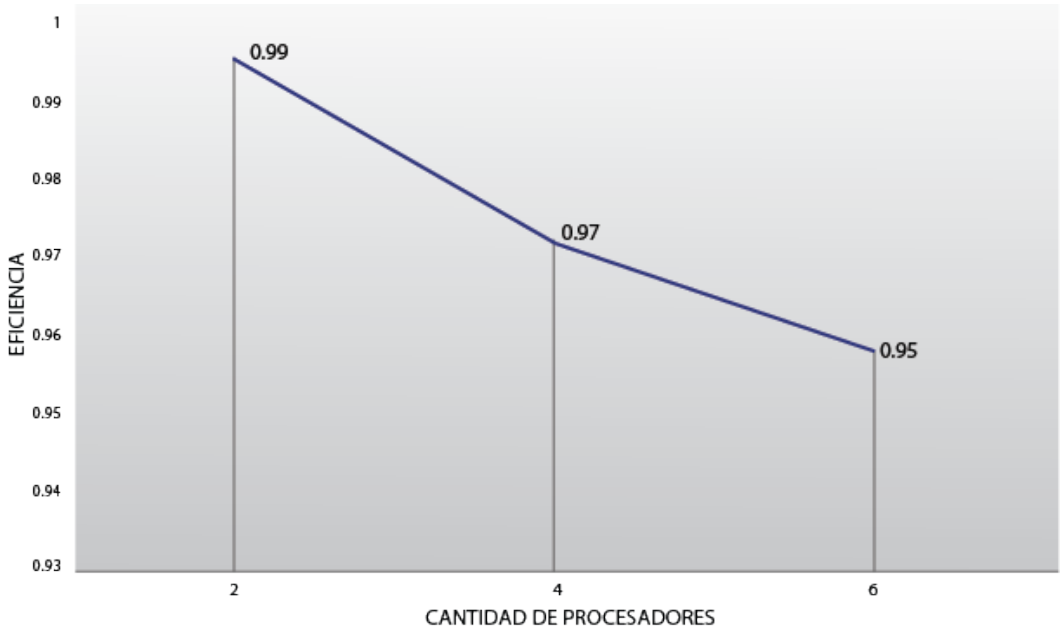

Figura 5.4: Eficiencia del AGP Maestro − Esclavo.

Pruebas	**Cant. de Procesos**	**Speedup**	**Eficiencia (%)**
Prueba 1	2	1,97	98,27
Prueba 2	4	3,92	98,06
Prueba 3	6	5,88	98,02

Tabla 5.7: Eficiencia del AGP Multipoblacional.

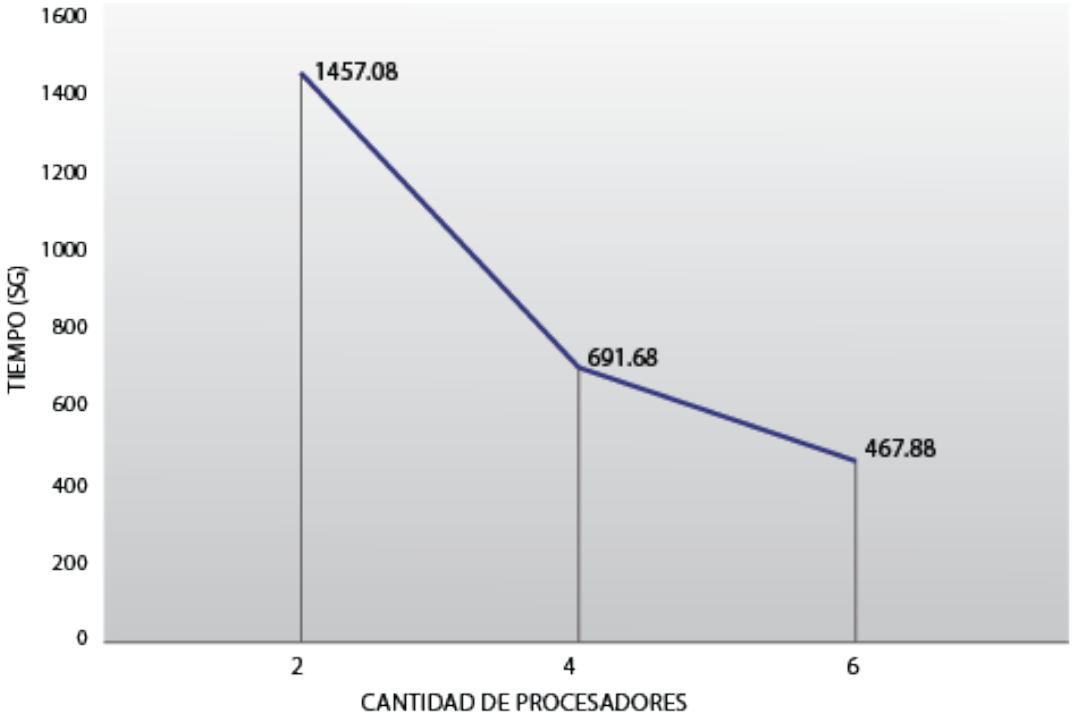

Figura 5.5: Tiempo de ejecución del AGP Multipoblación.

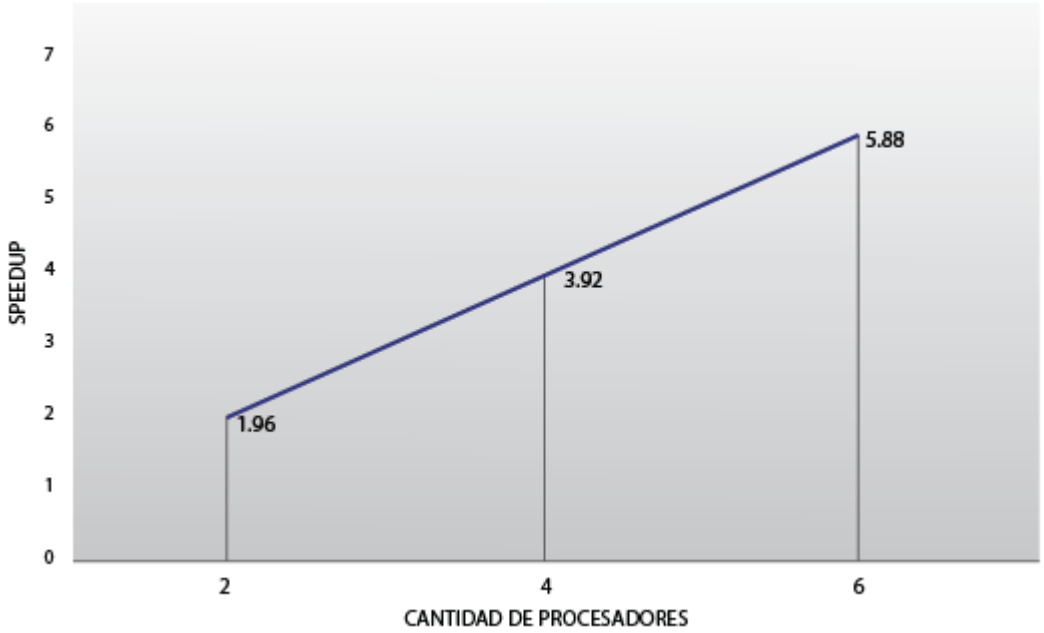

Figura 5.6: Valores de speedup del AGP Multipoblacional.

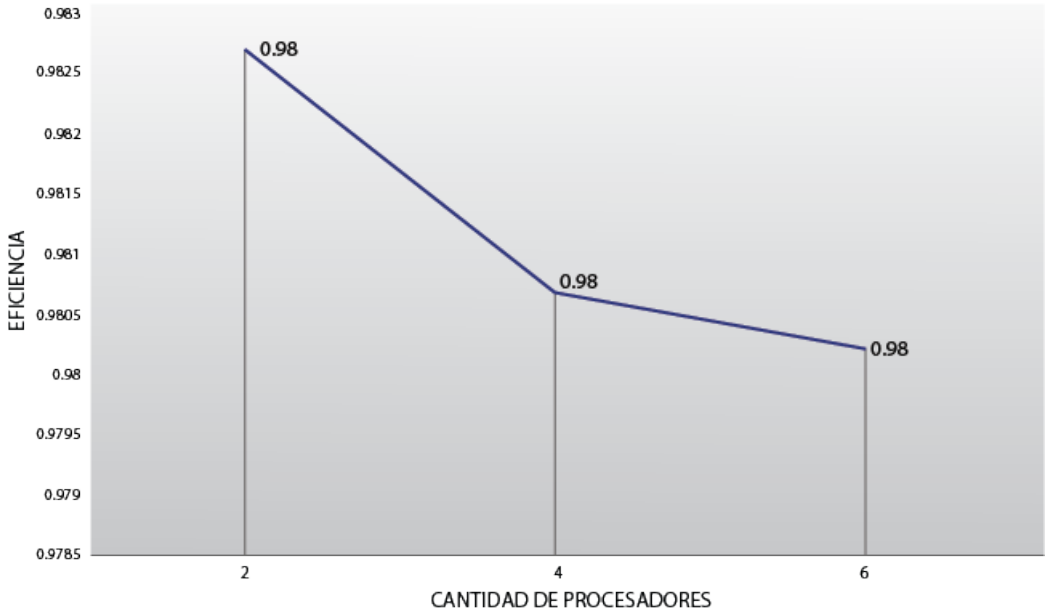

Figura 5.7: Eficiencia del AGP Multipoblacional.

5.3. Análisis de los resultados

La Tabla 5.8 muestra los resultados experimentales para el AG secuencial y el AGP Maestro − Esclavo, donde Fmax es el máximo Fitness (donde Fmax = $\min\{\zeta_j\}$, y $j \in \{1, 2, \ldots, N\}$ obtenido al final de la ejecución de los algoritmos, también se indica el tiempo de ejecución calculado por la media de 2 ejecuciones diferentes, Sp es el speedup calculado para los procesadores P, y E (%) representa la eficiencia paralela respectiva al tiempo, expresada en porcentaje.

AG Secuencial				
P	Fmax	Tiempo (s)		
1	15,41	2755,63		
AGP Maestro - Esclavo				
P	Fmax	Tiempo (s)	Sp	E(%)
2	15,57	1384,20	1,99	99,54
4	15,55	708,44	3,89	97,24
6	15,37	479,44	5,75	95,79
AGP Multipoblacional				
P	Fmax	Tiempo (s)	Sp	E(%)
2	15,72	1400,08	1,97	98,27
4	16,55	701,50	3,92	98,06
6	15,68	467,88	5,88	98,02

Tabla 5.8: Resultados para AG Secuencial, AGP Maestro - Esclavo y AGP Multipoblacional.

Los números en la columna P, en la Tabla 5.8, asociados a los resultados del algoritmo AGP Maestro − Esclavo corresponden a la cantidad de procesadores Esclavos disponibles en el cluster.

Se puede observar a partir de la Tabla 5.8 la reducción en el tiempo computacional cuando se aplicó el AGP Maestro − Esclavo comparando con la versión secuencial del AG, los valores del speedup (Sp) y para la eficiencia de cálculo paralelo, E (%), que representan un comportamiento sub-lineal para el AGP Maestro − Esclavo.

La aplicación del AGP Maestro − Esclavo produce una reducción sustancial del tiempo de cálculo, y su eficiencia es aproximadamente del 96 % (Fmax = 15,57, es la mejor solución para AGP Maestro − Esclavo); y el AGP Multipoblacional presenta una reducción similar en el tiempo de cálculo respecto al AG secuencial, sin embargo el AGP Multipoblacional obtuvo

mejores resultados que el AGP Maestro − Esclavo (Fmax = 16,55 % es la mejor solución para AGP Multipoblacional).

A partir de los resultados numéricos, mostrados en la Tabla 5.8, se selecciona el resultado de ajuste obtenido por el AGP Multipoblacional, correspondiente al mejor resultado de ajuste que otros algoritmos.

Las tablas 5.9 y 5.10 muestran los resultados numéricos de los parámetros del controlador, obtenidos por el AGP Multipoblacional.

#G	$\mathbf{K}_A$	$\mathbf{T}_A$	$\mathbf{K}_S$	$\mathbf{T}_1$	$\mathbf{T}_2$
30	100	0,08	19	0,10	0,001
31	100	0,10	16	0,09	0,001
32	100	0,05	17	0,07	0,010
33	100	0,05	17	0,09	0,001
34	100	0,02	19	0,05	0,019
35	100	0,05	20	0,10	0,008
36	100	0,07	15	0,10	0,007
37	100	0,05	18	0,10	0,007
38	100	0,05	20	0,06	0,001

Tabla 5.9: Valores de los parámetros para AVR y PSS.

#B	$\mathbf{K}_C$	$\mathbf{T}_C$
6	**100,0**	**0,15**
7	**138,3**	**0,40**
20	**128,3**	**0,02**

Tabla 5.10: Valores de los parámetros para SVC.

Los símbolos #G y #B, en columnas de las Tablas 5.9 y 5.10, representan el bus donde está conectado el generador correspondiente y el dispositivo SVC, respectivamente.

La figura (Fig. 5.8) muestra los mapas de valores propios de lazo abierto y lazo cerrado correspondientes a los mejores ajustes obtenidos por el método AGP Multipoblacional para la 2^{da} y 3^{ra} condicione de operación, donde los polos marcados con el símbolo $'0'$ y $'X'$ corresponden al circuito abierto y la situación de bucle cerrado, respectivamente.

Los valores de ajuste por el AGP Multipoblacional representan un alto coeficiente de amortiguación como se ilustra en la figura (Fig. 5.8). Las figuras (Fig. 5.9) y (Fig. 5.10) muestran la respuesta del ángulo del rotor de los

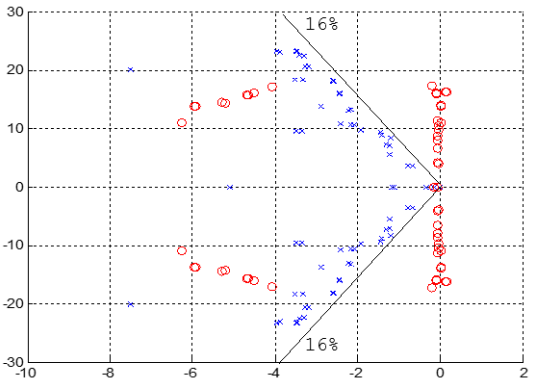

Figura 5.8: Posicionamiento de polos antes y después del ajuste coordinado.

generadores situados en los buses 33 y 38, la respuesta de tensión en los buses 4 y 16 respectivamente asociados a el segundo escenario operativo.

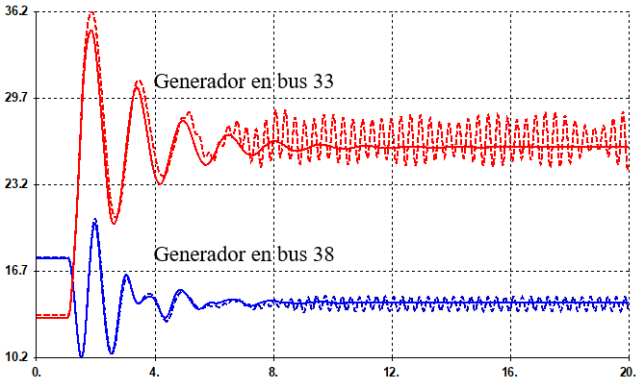

Figura 5.9: Respuesta del ángulo del rotor para la 2^{da} condición de operación.

En la figura (Fig. 5.9), las líneas discontinuas corresponden a la respuesta de tiempo no amortiguada del ángulo del rotor y las líneas continuas están asociadas a la respuesta de tiempo amortiguada de los ángulos del rotor, considerando la segunda condición de funcionamiento.

De la misma manera, las líneas discontinuas de la figura (Fig. 5.10) se refieren a la respuesta de tiempo de voltaje sin ningún dispositivo de compensación en esas subestaciones, y las líneas continuas corresponden a la respuesta de tiempo de tensión considerando los dispositivos SVC localizados en dichas subestaciones.

Las figuras (Fig. 5.11) y (Fig. 5.12) muestran la respuesta del ángulo

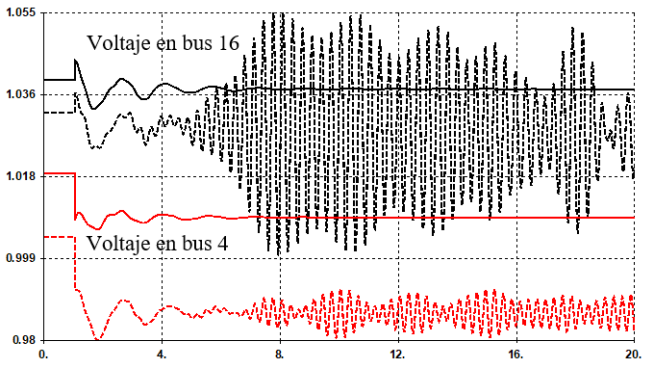

Figura 5.10: Respuesta de voltaje para la 2^{da} condición de funcionamiento.

del rotor de los generadores situados en los buses 33 y 38, y la respuesta del voltaje en los buses 3 y 25, respectivamente, asociados al tercer escenario operativo. Las líneas discontinuas de la figura (Fig. 5.11) representan la respuesta de tiempo no amortiguada y las líneas continuas de la respuesta amortiguada. Las respuestas de voltaje del SEP sin ningún dispositivo SVC están representadas por líneas discontinuas en la figura (Fig. 5.12).

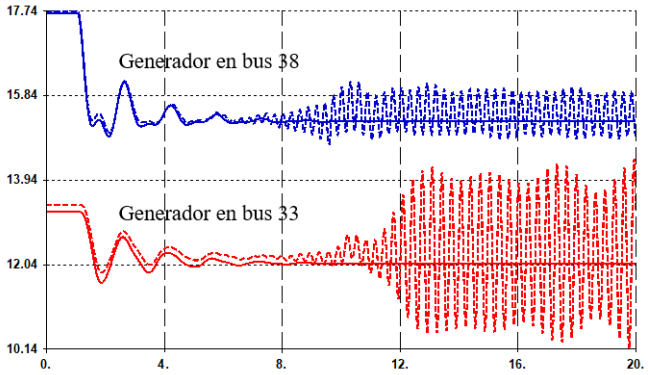

Figura 5.11: Respuesta del ángulo del rotor para la 3^{ra} condición de funcionamiento.

Las simulaciones dinámicas se compararon con la respuesta de generadores sin PSS, y no hay dispositivos SVC en ninguna subestación. Los parámetros AVR en esa situación se fijaron en $K_A = 200$ pu, y $T_A = 0{,}05$ s.

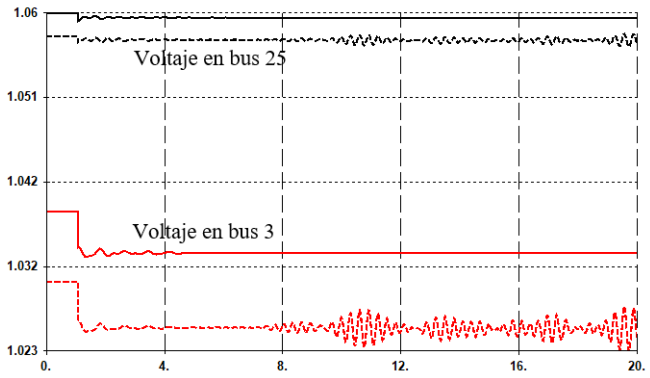

Figura 5.12: Respuesta del voltaje para la 3^{ra} condición de funcionamiento.

Capítulo 6

Conclusiones

El tiempo computacional del AG secuencial puede llegar a ser restrictivo para su aplicación en SEP de gran escala. Para ello, se implementó computación paralela al AG para reducir el tiempo computacional. Se desarrollaron dos algoritmos paralelos: AGP Maestro − Esclavo y AGP Multipoblacional. El AGP Maestro − Esclavo obtiene un Fitness similar al obtenido por el AG secuencial pero con un tiempo de computación reducido. Por otro lado, el AGP Multipoblacional no solo reduce el tiempo de computación sino que también obtiene, en promedio, valores de Fitness más altos. Ambos algoritmos se ejecutaron en computadoras personales interconectadas a través de una red de comunicación la cual resulta ser una solución económica y el ambiente paralelo es adecuado para este tipo de aplicación.

6.1. Logros alcanzados

- Implementar computación paralela síncrona del tipo Maestro - Esclavo al algoritmo genético meta-heurístico de ajuste coordinado de controladores de sistemas eléctricos secuencial.
- Implementar computación paralela síncrona del tipo Multipoblacional al algoritmo genético meta-heurístico de ajuste coordinado de controladores de sistemas eléctricos secuencial.
- Validar las implementaciones computacionales de los algoritmos genéticos paralelizados del tipo Maestro - Esclavo y Multipoblacional mediante ejecuciones en un cluster.
- Comparar los resultados obtenidos del algoritmo genético secuencial con las implementaciones computacionales paralelas del tipo Maestro - Esclavo y Multipoblacional ejecutados en un cluster de computadoras.

6.2. Solución del problema de investigación

Ha sido posible implementar computación paralela al problema de ajuste coordinado de controladores del SEP. Los resultados obtenidos durante las pruebas fueron satisfactorios, ya que se ha logrado disminuir el tiempo de ejecución, se pudo obtener un alto índice de aprovechamiento, y mejores resultados.

6.3. Sugerencias para la continuación del trabajo

- Implementar Computación Paralela de comunicación asíncrona para el AG de ajuste coordinado de controladores del SEP.
- Agregar más computadores al cluster, de tal forma a conseguir aumentar la escabilidad del cluster de computadoras, con el objetivo de reducir aun más el tiempo computacional y mejorar los resultados numéricos.
- Implementar otras estrategias de comunicación entre procesadores, y otras estrategias de recepción de datos, determinando posibles mejoras en la eficiencia del AGP del tipo multipoblacional.
- Comprobar la aplicabilidad de esta metodología, basada en la computación paralela, en sistemas eléctricos de mucho mayor porte.

Glosario

cluster Conjunto de computadoras interconectadas. 2, 3 , 5, 7 , 9, 10 , 53

síncrono Eventos que estan coordinados en el tiempo. 5

síncrona Eventos que estan coordinados en el tiempo. 5 , 10 , 15, 16 , 23 , 53

socket Software que funciona como punto final de las comunicaciones entre computadoras. 11

Referencias bibliográficas

[1] Antonio Luis Bergamo do Bomfim, Ajuste coordinado de estabilizadores de sistemas de potencia usando algoritmos genéticos. Tesis de doctorado en ciencias en Ingeniería Eléctrica Univeridad federal de rio de janeiro Acceso: 13 de noviembre de 2016.

[2] Manuel Sosa Rios, Enrique R. Chaparro Viveros, Ajuste Coordinado de Controladores de Sistemas Eléctricos de Potencia para Mejora de la Estabilidad Angular y de Tensión usando Algoritmos Genéticos. En línea: `http://www.laccei.org/LACCEI2013-Cancun/RefereedPapers/RP277.pdf` Acceso: 10 de noviembre de 2016.

[3] William D. Stevenson, Análisis de sistemas eléctricos de potencia. En línea: `http://blog.espol.edu.ec/econde/files/2012/08/analisis_de_sistemas_electricos_de_potencia_stevenson_.pdf` Acceso: 15 de noviembre de 2016.

[4] Algoritmos Genéticos. En línea: `http://eddyalfaro.galeon.com/geneticos.html` Acceso: 11 de abril de 2016.

[5] Inteligencia Artificial. En línea: `http://es.slideshare.net/uni_fcys_sistemas/desarollo-de-sofware-con-algoritmos-genticos` Acceso: 13 de mayo de 2016.

[6] GNU Octave. En línea: `https://www.gnu.org/software/octave/about.html` Acceso: 11 de febrero de 2016.

[7] Miguel Vargas, Cómputo en paralelo con MPI. En línea: `personal.cimat.mx/~miguelvargas/Coursenotes/ComputoenparaleloconMPI.pdf` Acceso: 04 de abril de 2016.

[8] M. Creel, PelicanHPC Tutorial. En línea: `https://www.scribd.com/doc/54531563/PelicanHPC-Tutorial-en-espanol` Acceso: 20 de octubre de 2015.

[9] MPI: A Message-Passing Interface Standard. En línea: http://mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf Acceso: 16 de noviembre de 2016.

[10] Ajuste Coordinado de Estabilizadores de Sistemas de Potencia y Reguladores Automáticos de Tensión utilizando Algoritmos Genéticos Multi Objetivos. En línea: http://www.une.edu.py:82/fpune_scientific/index.php/fpunescientific/article/viewFile/101/102

[11] M. Creel, Econometrics. En línea: http://pareto.uab.es/mcreel/Presentations/ParallelEconometrics.pdf Acceso: 11 de abril de 2016.

[12] PelicanHPC GNU Linux. En línea: http://pareto.uab.es/mcreel/PelicanHPC/ Acceso: 05 de febrero de 2016.

[13] Programacion Paralela en MPI. En línea: http://pareto.uab.es/mcreel/Presentations/ParallelEconometrics.pdf Acceso: 11 de abril de 2016.

[14] GNU Octave En línea: https://www.gnu.org/software/octave/ Acceso: 07 de mayo de 2016.

[15] Análisis y modelado del rendimiento de algoritmos paralelos en cluster de computadoras En línea: http://www.saber.cic.ipn.mx/cake/SABERsvn/trunk/Repositorios/webVerArchivo/619/1 Acceso: 07 de mayo de 2016.

[16] IEEE Tutorial Course, Power System Stabilization via Excitation Control, 81 EHO 175-0 PWR, 1980.

[17] Programación Dinámica Paralela de un cluster HPC. En línea: https://goo.gl/6L9oN7 Acceso: 20 de junio de 2016.

[18] P. Kundur, Power System Stability and Control, Mc.Graw-Hill, 1993.

[19] A.L.B. do Bomfim, G.N. Taranto, and D.M. Falcão, Simultaneous Tuning of Power System Damping Controllers Using Genetic Algorithms, IEEE Transactions on Power Systems, vol. 15, no. 1, pp. 163-169, February 2000.

[20] Francisco Javier Bris Peñalver, Cómo medir la eficiencia en algoritmos paralelos En línea: https://es.scribd.com/document/56700301/Eficiencia-en-Algoritmos-Paralelos-Bris Acceso: 03 de mayo de 2017.

[21] Ing. Luis Alberto Rivera Zamarripa, Análisis y modelado del rendimiento de algoritmos paralelos en clusters de computadoras. En línea: http://tesis.ipn.mx/bitstream/handle/123456789/16211/XM%2012.20.pdf Acceso: 31 de agosto de 2015.

[22] Revista de Tecnología e Innovacion 2015, Liebres Inteligentes: Sistema de multipomputadoras para el procesamiento paralelo de aplicaciones científicas. En línea: http://www.ecorfan.org/bolivia/researchjournals/Tecnologia_e_innovacion/vol2num3/Revista-de-Tecnologia-e-Innovacion-vol-3-122-131.pdf Acceso: 03 de mayo de 2017.

[23] Open MPI: Open Source High Performance Computing. En línea: http://www.open-mpi.org Acceso: 10 de junio de 2016.

[24] Gregory G. Howes, Department of Physics and Astronomy. University of Iowa, Parallel Programming Using MPI 2012. En línea: http://www2.physics.uiowa.edu/~ghowes/teach/ihpc12/lec/ihpc12Lec_MPI_IHPC12.pdf Acceso: 10 de junio de 2016.

[25] Algoritmos Genéticos. En línea: http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/temageneticos.pdf Acceso: 04 de mayo de 2017.

[26] Fanny González, Luís Fariña, Eustaquio Martńez, Esteban Vargas y Anastacio Arce, Lagrangean Relaxation Parallel Method for Optimizing of a Hydroelectric Generation System. En línea: http://www.sciencedirect.com/science/article/pii/S1571066111001770. Acceso: 09 de octubre de 2017.

[27] Academic work that uses PelicanHPC or ParallelKnoppix. En línea: http://www.pelicanhpc.org/academic_work.html. Acceso: 09 de octubre de 2017.

[28] R. T. Byerly, D. E. Sherman and R. J. Bernnon, Frequency domain analyisis of Low frequency oscillations in large electric systems. Report EPRI EL, 1978.