77

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

"ALGORITMOS GENETICOS PARALELOS EN ESTIMACION ESPECTRAL DE SEÑALES DOPPLER"

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERA EN COMPUTACION

P R E S E N T A:

KATYA RODRIGUEZ VAZQUEZ

DIRECTOR DE TESIS: DR. JULIO SOLANO GONZALEZ

MEXICO; D. F. 1994

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

DEDICATORIA

A mis padres Antonio y Beatriz y a mis hermanas Susana y Marisa.

AGRADECIMIENTOS

La autora agradece a la Universidad Nacional Autónoma de México y la Dirección General de Intercambio Académico (Proyecto PAPIID-D0303593) por su soporte financiero y al Consejo Nacional de Ciencia y Tecnología por el financiamiento del Sistema Paralelo basado en Transputers, fundamental en el desarrollo de este trabajo (Proyecto PACIME-F284-A9209).

PUBLICACIONES

El trabajo presentado en esta tesis ha generado información publicada en los siguientes artículos:

[1] Solano, J. y García Nocetti, D.F.,*Implementation of a Parametric Spectral Estimator using Genetic Algorithms,* IEE International Conference on **Control 94**, University of Warwick, Coventry, marzo 1994, pp. 754-759.

[2] Solano, J. y Rodríguez K.,*Determinación de Parámetros y Métodos de Selección en Algoritmos Genéticos,* XVI Congreso Académico Nacional de Ingeniería Electrónica, ELECTRO 94, Instituto Tecnológico de Chihuahua, octubre 1994, pp. 616-626.

[3] Solano, J. y Rodríguez K., *Implementación de un Estimador Espectral Paramétrico Mediante Algoritmos Genéticos*, XVI Congreso Académico Nacional de Ingeniería Electrónica, ELECTRO 94, Instituto Tecnológico de Chihuahua, octubre 1994, pp. 484-494.

[4] Solano, J. y Rodríguez K., *Algoritmos Genéticos en Estimación Espectral de Señales Doppler Ultrasónicas*, Conferencia Internacional, Ciencia y Tecnología para el Desarrollo, La Habana, Cuba, a celebrarse en enero de 1995. (Artículo Aceptado).

[5] Solano, J., García Nocetti, D. y Rodríguez K., *Parallel Genetic Algorithms in Spectral Estimation of Doppler Signals*, 3rd. IFAC Workshop on Algorithms and Architectures for Real-Time Control, Ostende, Bélgica, a celebrarse en mayo de 1995. (Artículo Sometido).

CONTENIDO

CAPITULO I

INTRODUCCION.

I.1. DESCRIPCION GENERAL.

I.1.1. Motivación.

Diversas técnicas que utilizan ultrasonido Doppler son actualmente utilizadas con mayor frecuencia, con el objeto de valorar enfermedades circulatorias en forma no invasiva, proporcionando con ésto medios poderosos para el diagnóstico y monitoreo de arterias que presentan problemas de estenosis.

En general, los instrumentos de ultrasonido Doppler detectan el movimiento de un objeto que interfiere la trayectoria del haz ultrasónico a través del corrimiento Doppler de la señal dispersada o reflejada por dicho objeto. En nuestro caso, dichos objetos son células de sangre, las cuales bajo condiciones patológicas alteran sus patrones de velocidad. Estos cambios usualmente contienen información muy útil. Adicionalmente, este tipo de lesiones provocan turbulencias en el flujo que resultan en un incremento en el rango del corrimiento de frecuencia Doppler (espectro más ancho). Idealmente, los métodos de análisis de señales Doppler deben considerar estos importantes aspectos.

Típicamente la literatura se refiere a dos tipos de dispositivos de ultrasonido Doppler: de onda continua y de onda pulsada. Los dispositivos de onda continua poseen una gran desventaja, es imposible distinguir estas señales reflejadas en las arterias a diferentes niveles, lo cual puede solucionarse mediante el uso de dispositivos de onda pulsada.

El análisis de señales Doppler involucra, generalmente, una secuencia de procedimientos, desde la recopilación de datos clínicos, hasta la aplicación e interpretación del espectro Doppler estimado. Estas etapas introducen inevitablemente errores de medición, muchos de los cuales son detectados y anulados al aplicar técnicas adecuadas de análisis espectral, tales como el uso de ventanas y promedios espectrales.

Técnicas convencionales de análisis de señales Doppler utilizan la Transformada Rápida de Fourier sobre ventanas de la señal en forma secuencial o traslapada. Sin embargo, las limitaciones de estas técnias para estimar el espectro Doppler ha sido dado a conocer en

forma extensa.

Otra clase de métodos de estimación espectral comprenden los denominados métodos **paramétricos**. Su gran ventaja es su mejor resolución en frecuencia, mientras que una desventaja que presenta es su complejidad computacional.

No obstante, la gran variedad de técnicas como las descritas, se siguen llevando a cabo muchos trabajos para mejorar diversos aspectos de estimación espectral. Tal es el caso del método alternativo presentado en esta tesis.

Por otra parte, en la actualidad muchos de los desarrollos en el campo de la Computación se basan en mecanismos naturales, tratando de emular básicamente dos esquemas: la estructura cerebral y la evolución genética. Dichos esquemas han sido fundamentales para el desarrollo de nuevas y mejores arquitecturas de hardware y algoritmos en software, como es el caso de los **Algoritmos Genéticos**, los cuales son herramientas básicas en el desarrollo de este trabajo.

De manera general, los **AG's** son procedimientos de búsqueda basados en los mecanismos de la evolución natural que involucran un intercambio estructurado de información aleatoria. Esto da por resultado la supervivencia de los individuos más aptos entre una población de posibles soluciones del problema en estudio. Estos algoritmos usan representaciones simples para codificar estructuras complejas y mejorarlas por medio de transformaciones elementales.

Una gran mayoría de lenguajes de programación convencionales operan en forma secuencial; ésto derivado de los esquemas de arquitecturas secuenciales de las computadoras convencionales (basadas en los conceptos de Von Neumann), las cuales utilizan un solo procesador, memoria y dispositivos de entrada/salida ligados a un bus de datos. Sin embargo, muchos problemas y algoritmos presentan un **paralelismo inherente**, el cual puede ser explotado eficientemente mediante el uso de técnicas adecuadas.

La naturaleza misma de los **Algoritmos Genéticos** permite que éstos sean métodos apropiados para el uso de técnicas de **Procesamiento Paralelo**. Con el desarrollo en los últimos años de arquitecturas paralelas y de la **multiprogramación**, tenemos la posibilidad de realizar implementaciones eficientes de algoritmos que presentan un procesamiento intensivo, pudiéndose éstos ser ejecutados en **tiempo real**.

Notables desarrollos como el **transputer** de Inmos y su lenguaje asociado **OCCAM**, pueden considerarse como un soporte importante en la aplicación efectiva y a bajo costo de procesamiento paralelo.

El objetivo general del trabajo presentado en esta tesis es el poder utilizar eficientemente el paralelismo inherente que los **Algoritmos Genéticos** presentan, al explorar simultáneamente diversas regiones del espacio de soluciones y al explotar la independencia de los procedimientos propios del algoritmo. Esto para mejorar tanto la calidad de su respuesta como su tiempo de ejecución, aprovechando para ello las ventajas asociadas al

transputer y el modelo de paralelismo de **OCCAM.**

El trabajo presenta la implementación de un estimador espectral paramétrico empleando esta nueva alternativa para su solución con la finalidad de reducir la complejidad computacional del algoritmo convencional haciendo uso de la simplicidad asociada a esta solución genética. Esta implementación ha sido propuesta para ser usada por un analizador espectral Doppler Ultrasónico, el cual calculará y desplegará, en tiempo real, el contenido de frecuencia de señales de ultrasonido Doppler que puede ser usado en aplicaciones de instrumentación médica.

I.2. CONTENIDO DE LA TESIS.

Para efectos de estudio y desarrollo el trabajo se encuentra dividido en los siguientes capítulos:

Capítulo I. Se presenta una introducción general, enfatizando las razones que motivaron este trabajo. Incluye también el contenido de la tesis, así como sus principales contribuciones.

Capítulo II. En este capítulo se presentan la teoría y los conceptos de los principales temas involucrados en la implementación del Estimador Espectral Paramétrico aplicado en Flujometría Doppler. Los temas expuestos son: *Conceptos y teoría de los Algoritmos Genéticos, fundamentos de Procesamiento Paralelo incluyendo el concepto de Transputer y el lenguaje de programación OCCAM, y las bases de los Métodos de Estimación Espectral Convencionales y Paramétricos.*

Capítulo III. Esta sección expone un análisis de diversos métodos de selección y el efecto de diferentes parámetros en el comportamiento de los Algoritmos Genéticos. Este análisis está realizado en el intérprete MATLAB.

Capítulo IV. Basados en el estudio realizado en el capítulo III, este capítulo presenta la implementación del método de estimación espectral paramétrico de Covariancia Modificada mediante el uso de Algoritmos Genéticos. Asimismo se presenta un estudio comparativo de la respuesta espectral entre la implementación genética y su solución convencional.

Capítulo V. El desarrollo del estimador paramétrico ha sido propuesto para ser aplicado en Flujometría Doppler. En este capítulo se presentan los conceptos básicos del ultrasonido y señales Doppler, además de un estudio y análisis de las respuestas presentadas por la solución genética y los resultados proporcionados por el sistema de ecuaciones de la matriz de covariancia, para esta aplicación en particular.

Capítulo VI. Con la disponibilidad de una plataforma de transputers y el desarrollo de programas en el lenguaje de programación OCCAM, se presenta la implementación de tres modelos paralelos de Algoritmos Genéticos, realizando una evaluación de los tiempos de ejecución y eficiencia de estos modelos. Se presenta también una análisis de las ventajas que

estos modelos pueden ofrecer.

Capítulo VII. En este capítulo se presentan las conclusiones finales del trabajo de tesis realizado y las perspectivas futuras del mismo.

Esta tesis presenta información adicional en cuatro apéndices:

En el Apédice A se tiene el código del Algoritmo Genético Secuencial implementado en el capítulo III.

El Apéndice B corresponde a las rutinas adiconales al Algoritmo Genético Secuencial del Apéndice A con lo cual se tiene la alternativa genética del estimador espectral.

El Apéndice C contiene una descripción del lenguaje de programación OCCAM, una descripción general de la plataforma de transputers y la configuración de los mismos en una determinada topología, así como las instrucciones requeridas para la ejecución de un programa determinado.

En el Apéndice D se presenta el pseudocódigo de las implementaciones paralelas de los tres modelos de Algoritmos Genéticos y los archivos de configuración de red de transputers de las topologías propuestas en los modelos paralelos.

I.3. APORTACIONES.

El método genético presentado como una solución alternativa proporciona una gama de posibilidades para la búsqueda de soluciones óptimas de problemas generales.

La aplicación de este método al análisis espectral de señales Doppler utilizando métodos paramétricos, presenta herramientas para reducir la complejidad computacional del estimador espectral paramétrico, además de mejorar la respuesta en frecuencia del espectro de la señal Doppler en comparación con métodos de estimación espectral convencionales, abriendo así la posibilidad para la implementación de otro tipo de métodos espectrales aun más complejos que contribuyan en una mejor estimación espectral.

La disponibilidad de sistemas paralelos permite la explotación eficiente del paralelismo inherente que presentan los Algoritmos Genéticos, tanto para disminuir el tiempo de ejecución como para mejorar la calidad de las soluciones generadas. Esto nos permite abordar problemas más complejos con restricciones específicas (ej. tiempo real).

CAPITULO II

ANTECEDENTES

II.1. INTRODUCCION.

El propósito de este capítulo es proporcionar al lector los antecendentes esenciales de las áreas principales que envuelve el desarrollo del proyecto presentado en esta tesis y de este modo ayudar a que los tópicos descritos en los capítulos subsecuentes puedan ser mejor apreciados.

Los temas que se presentan, en este capítulo, son: conceptos y teoría de los Algoritmos Genéticos, fundamentos de Procesamiento Paralelo para el desarrollo de alternativas de algoritmos paralelos en base a la implementación genética (incluyendo el concepto de Transputer y las características básicas del lenguaje de programación OCCAM), y Métodos de Estimación Espectral Convencional y Paramétrica.

II.2. ALGORITMOS GENETICOS.

En los últimos años, la Ciencia de la Computación ha dirigido muchos de sus estudios hacia los sistemas biológicos y naturales. Esto se ha tomado básicamente a partir de dos esquemas naturales de aprendizaje: el cerebro y la evolución [1]. Estudios de la estructura cerebral han proporcionado fuentes de nuevas ideas para arquitecturas computacionales. Mientras que la evolución y selección natural han generado grandes desarrollos como los algoritmos genéticos y los programas evolutivos.

II.2.1. DEFINICION.

Los *Algoritmos Genéticos (AG's)* son procedimientos de adaptación los cuales están basados en mecanismos de evolución natural (ej. la teoría hereditaria y la evolución en el campo de poblaciones genéticas). Combinan la supervivencia de los individuos más aptos de una población, con operadores genéticos tomados de la naturaleza, para formar un mecanismo robusto

que puede ser adaptado a una gran variedad de problemas [14][16].

Los Algoritmos Genéticos combinan la explotación de soluciones conocidas con la exploración de nuevas áreas del espacio de posibles soluciones. La eficacia de los Algoritmos Genéticos dependen de una apropiada mezcla o combinación entre la explotación y la exploración; la selección de acuerdo a las características y aptitudes de los individuos de una población, constituye la base de la explotación, mientras que, los operadores de mutación y cruzamiento son la base de la exploración [10].

Un AG simula el comportamiento dinámico de una población genética generando una base de conocimientos formada por estructuras que se desenvuelven o desarrollan en respuesta al desempeño observado en su medio ambiente operacional. Cada estructura es presentada al algoritmo como una secuencia de sus componentes constitutivos y es manipulada por operadores genéticos. Las estructuras de una población se van adaptando al medio en el que se desenvuelven; ésto es, se van modificando de acuerdo a diferentes condiciones (ej. probabilidades de selección, probabilidad de contribuir a la formación de un nuevo miembro de la población siguiente, etc.). La adaptación involucra una modificación progresiva de alguna o algunas estructuras de la generación presente.

En términos generales, los Algoritmos Genéticos están constituídos por:

- El cálculo de las condiciones de cada individuo de la población.
- La selección de individuos para producir los descendientes de la siguiente generación, y
- La reproducción de dichos individuos por medio de operadores genéticos.

En esta sección se presenta un panorama general de las principales características de estos algoritmos, incluyendo una breve historia de los mismos, la definición de un *Algoritmo Genético Simple (AGS)*, sus operadores básicos y la teoría en la que éste se basa *(Teorema del Esquema)*.

II.2.2. HISTORIA DE LOS ALGORITMOS GENETICOS.

A finales de los años 50's y principios de la década de los 60's se introdujeron las bases de los Algoritmos Genéticos. Una de las primeras aportaciones fue el denominado **Esquema de Box** [5] (1957) de **Operación Evolutiva**. Esta fue más que un algoritmo una técnica de administración. Dicho esquema fue herramienta muy útil y antecesor de nuevas técnicas de búsqueda, aunque no fue un algoritmo genético en sí.

Después del trabajo de **Box**, surgieron numerosas técnicas de optimización basadas en la evolución, tales son los casos de Blendsoe (1961), Bremermann (1962) y Friedman (1959), por mencionar algunos. Los estudios de Blendsoe y Bremermann vinieron a concluir en el concepto

moderno de un algoritmo genético. Ambos sugirieron una codificación binaria. Blendsoe presentó un esquema que combinaba la generación de individuo por individuo, mutación y así obtenía la mejor selección. Además propuso un esquema de población por población. Bremermann extendió el trabajo de Blendsoe para la generación de poblaciones sucesivas de cadenas usando selección y mutación, y también propuso un nuevo operador de recombinación.

Por otra parte, algunas Técnicas de **Estrategias de Evolución** fueron desarrolladas en la Universidad Tecnológica de Berlín por Rechenberg (1965). Los experimentos de Rechenberg desarrollaron una figura de plano aerodinámico usando un aparato físico que permitía perturbación de la geometría del plano.

La operación evolutiva y las técnicas de optimización evolutivas fueron seguidas por las técnicas de **Programación Evolutiva** de Fogel, Owens y Walsh (1966). En este trabajo una variedad de tareas predictivas de símbolos secuenciales fueron ejecutadas por búsquedas a través de un espacio de pequeñas máquinas de estado finito [13]. Este tipo de máquina fue enseñada para predecir ciclos repetitivos de símbolos de salida usando las técnicas de Fogel, Owens y Walsh de programación evolutiva, las cuales inicialmente consistían de dos operadores: **selección y mutación.** Para Fogel, Owens y Walsh, la mutación era una simple modificación del diagrama de estado de la máquina de estado finito.

En esta misma década, la teoría fundamental de los algoritmos genéticos fue expuesta por vez primera. De acuerdo a John H. Holland [19]:

> " El estudio de la adaptación involucra el estudio del sistema adaptable y su medio. En términos generales, es un estudio de cómo los sistemas pueden generar procedimientos con la capacidad de ajustarse eficientemente a las características propias de su medio. Si la adaptabilidad no es arbitrariamente restringida al inicio, los sistemas de adaptación deben ser capaces de generar algún método o procedimiento capaz de una definición efectiva. "

La finalidad de Holland fue el desarrollo de la teoría y procedimientos necesarios para la creación de programas generales con capacidades ilimitadas para adaptarse a medios arbitrarios. En ese mismo tiempo, Holland reconoció el papel fundamental de la selección no natural, la sobrevivencia del más apto en forma artificial. Holland solo sugirió la importancia del crossover (cruzamiento) y otros operadores genéticos de recombinación. El primer escrito sobre estos operadores fue realizado en 1965.

Este fundamento teórico y el papel fundamental de la recombinación (John H. Holland, 1965), fueron las bases para la creación de la **Teoría del Esquema** (John H. Holland, 1968-1971) [14][19]. De acuerdo a este autor, un esquema es un modelo de similitudes que describe un subconjunto de cadenas con semejanzas en ciertas posiciones de la misma. La importancia de la **teoría del esquema** fue algo muy relevante en esa década con importantes resultados ligados a

los planes de reproducción.

La primera mención de las palabras "algoritmos genéticos" y su primera publicación se dio en la disertación de Bagley [2] en 1967. Bagley construyó algoritmos genéticos para buscar conjuntos de parámetros en las funciones de evolución de un juego y las comparó con algoritmos de correlación y procedimientos de aprendizaje. Bagley encontró que los algoritmos de correlación requerían una buena relación entre la no linealidad del juego y la no linealidad de los algoritmos de correlación. Bagley además introdujo los mecanismos de escalamiento.

En este mismo tiempo, Rosenberg [30] investigó los algortimos genéticos en su disertación doctoral. El hizo énfasis en los aspectos biológicos y de simulación, además de desarrollar un esquema de cruzamiento adaptativo.

1975 fue un año muy importante en el desarrollo de la teoría de los algoritmos genéticos. John H. Holland publicó su libro "**Adaptation in Natural and Artificial Systems**", y en este mismo año, De Jong contribuyó con su importante disertación, " Un Análisis del Comportamiento de una Clase de Sistemas Adaptables Genéticos ". El estudio de De Jong [9] es uno de los pilares del desarrollo de estos algoritmos, es la combinación de la **Teoría del Esquema** de Holland y de sus propios experimentos en cómputo.

Como en los estudios de Hollstein [19] (1971), De Jong [9] consideró el algoritmo genético en optimización de funciones pero también tuvo la visión del potencial de estos algoritmos en otras áreas como el estudio sobre diseño de estructuras de datos, diseño de algoritmos y sobre control adaptable de sistemas operativos. De Jong realizó también estudios sobre la importancia de la recombinación estructurada.

De Jong comparó los algoritmos genéticos con las técnicas de gradientes. En su estudio propuso cinco funciones que exhibían un rango de propiedades que presentaban dificultades en las técnicas de gradientes. Antes de realizar esta comparación, estudió la influencia del tamaño de la población y de los rangos de mutación y cruzamiento en la eficiencia de la búsqueda genética. También investigó el uso del cruzamiento generalizado o multipunto.

II.2.3. TERMINOLOGIA.

La terminología utilizada en la literatura presenta una mezcla de términos utilizados en genética natural y en ciencias de la computación. A continuación se presenta una tabla comparativa de términos naturales y términos empleados en los AG's.

Natural	Algoritmos Genéticos
cromosoma	cadena
gen	carácter o indicador
alelo	valor o estado del indicador
genotipo	estructura
fenotipo	conjunto de parámetros, solución alternativa, estructura decodificada

II.2.4. ALGORITMOS GENETICOS VS. OTROS METODOS.

Los AG's difieren de los métodos tradicionales de búsqueda y optimización en los siguientes puntos:

1. Los AG's tratan el problema como una caja negra. A diferencia de los métodos tradicionales que van a resolver un problema específico.
2. Los AG's usan codificación, no variables de decisión.
3. Los AG's buscan en una población de posibles soluciones, no se basan en un solo punto.
4. Los AG's emplean operadores aleatorios, no usan reglas determinísticas.

Muchos métodos de búsqueda requieren de ciertos antecedentes o información adicional del problema en estudio para llegar a las soluciones. Por ejemplo, los métodos de gradientes necesitan de derivadas para llegar a encontrar los valores óptimos. Otras técnicas requieren conocer la mayoría de los parámetros involucrados. En el caso de los AG's, éstos no necesitan de información adicional, tratan el problema como una caja negra y solo requieren evaluar la función objetivo.

En la práctica, los AG's codifican las posibles soluciones de un problema determinado como una cadena de longitud finita (una cadena de bits), realizando posteriormente, el procesamiento sucesivo de esta población de cromosomas artificiales en generaciones sucesivas. Al tener una representación codificada de las posibles soluciones, puede explotarse las similitudes que existan entre éstas. Esto le da a los AG's un amplio rango de aplicaciones.

La mayoría de los métodos tradicionales de búsqueda y optimización trabajan punto a punto, empleando el resultado de un punto para determinar cual será el siguiente punto a analizar. Esto puede llevarnos a valores óptimos falsos. Por el contrario, los AG's realizan aproximaciones de población a población en las cuales se presenta una probabilidad mucho menor de obtener

valores óptimos falsos.

También se presentan en los AG's el empleo de reglas de transición probabilísticas y no reglas determinísticas, como es el caso de la mayoría de los métodos tradicionales. Además, los AG's emplean una selección aleatoria para guiar la búsqueda hacia un espacio de soluciones con mayores probabilidades de incluir los valores óptimos.

II.2.5. ALGORITMO GENETICO SIMPLE.

Un **Algoritmo Genético Simple (AGS)** (Figura 2.2.1) está compuesto de un *esquema de reproducción* para la selección de los genotipos más aptos que se utilizarán en la creación de los descendientes de la nueva generación, y un conjunto de operadores genéticos.

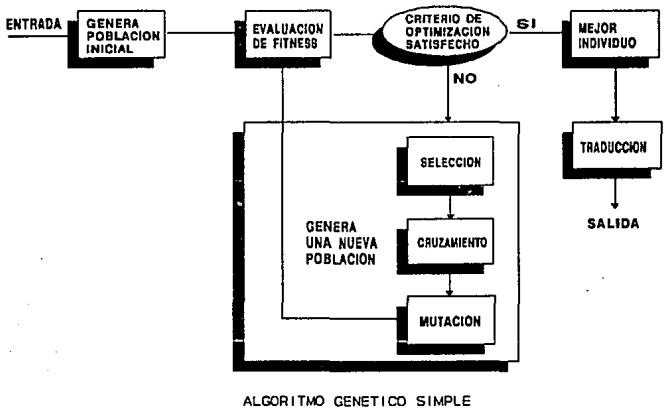


Figura 2.2.1. Algoritmo Genético Simple.

El esquema de reproducción parte de n estructuras de cadenas de m bits las cuales pueden ser generadas aleatoriamente. Cada estructura es evaluada y se le asigna una medida generalmente llamada *fitness* (una medida de su desempeño o adaptación al problema). Las probabilidades de selección son calculadas para cada estructura basándose en su valor particular de *fitness*. La nueva generación de estructuras es creada seleccionando estructuras de acuerdo a las probabilidades asignadas en la presente generación y con la aplicación a ésta de un conjunto de

operadores genéticos. Estos operadores se aplican a cada pareja tomadas de la población presente por el esquema de reproducción hasta que n nuevas estructuras son creadas. La base de conocimientos de la nueva generación es entonces reemplazada por la generación presente. Las nuevas estructuras son evaluadas y este ciclo se repite hasta un número previamente definido de generaciones o si algún criterio de terminación es alcanzado.

II.2.5.1. Esquemas de Reproducción.

Como se mencionó anteriormente, el esquema de reproducción es el que determina que individuos son los que cuentan con las condiciones más aptas para contribuir a la formación de nuevos descendientes. Se presentan numerosas formas para llevar a cabo la implementación de estos esquemas. Estas formas de implementación son también denominados métodos de selección. La fase de selección puede dividirse en dos algoritmos distintos [16]: el algoritmo de selección y el algoritmo de muestreo. Los algoritmos de selección asignan a cada uno de los individuos de una población un número real que corresponde al valor de la función objetivo. Este valor indica el número esperado de descendientes a ser generados de ese individuo en particular en el tiempo t. El algoritmo de muestreo produce una nueva población creando copias de los individuos basado en el porcentaje de muestras obtenido de la función objetivo.

Algunos de los métodos de selección comúnmente usados son los siguientes [15][17]:

- Reproducción proporcionada.
    * Determinístico.
    * Estocástico.
- Esquemas de ordenamiento *ranking* (Baker, 1985).
- Torneo.
    * Con substitución: elección aleatoria de dos individuos de la población, seleccionando el mejor.
    * Sin substitución: uso de n-permutaciones, reteniendo el par de estructuras en torneo, guardando la mejor.
- Genitor.

Reproducción Proporcionada. Este método describe un grupo de esquemas de selección que eligen individuos de la población para que pasen a la siguiente generación y se crucen de acuerdo a sus valores f de la función objetivo. En estos esquemas, la probabilidad de selección

P_i de un individuo del *i-ésimo* esquema H en la generación t es:

$$P(i,t)=\frac{f_i}{\sum_{j=1}^{k} m(j,t)f_j} \qquad (2.2.1)$$

donde k esquemas existen y n es el número total de individuos. Varios métodos se han sugerido para implementar esta distribución de probabilidad como: el método de la ruleta (Figura 2.2.2.), selección estocástica residual (Booker [4], 1982; Brindle [6], 1981), y la selección estocástica universal (Baker [3], 1987; Grefenstette y Baker [17], 1989).

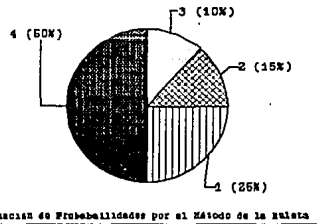


Figura 2.2.2. Esquema de Selección de la Ruleta.

El muestreo determinístico es un esquema donde las probabilidades de selección son calculadas como se muestra en la ecuación 2.2.1. Entonces, el número esperado de descendientes para cada cadena e_i es calculado como $e_i=P_i*n$. Cada cadena se le asignan muestras de acuerdo a la parte entera del valor de ei, y la población es ordenada de acuerdo a la parte fraccional del valor de ei. Las cadenas restantes necesarias para completar la población son sacadas del inicio de la lista ordenada.

Por otra parte, el método estocástico inicia de una manera idéntica al muestreo determinístico. Los valores de individuos o descendientes esperados son calculados en el momento y las partes enteras son asignadas. En los muestreos restantes estocásticos con substitución, las partes fraccionales de los valores esperados son usadas para calcular pesos en el procedimiento de selección de la ruleta que es usado para completar los espacios restantes. En los muestreos restantes sin substitución, las partes fraccionales de los valores esparados son tratados como probabilidades.

Esquemas de Ordenamiento. Este tipo de selección ordena la población de acuerdo al

valor de la función objetivo, asignando el número de copias que cada individuo va a recibir de acuerdo a un criterio previamente establecido. Posteriormente, se aplica la reproducción proporcionada de acuerdo a la asignación anterior.

Torneo. En este método se eligen individuos aleatoriamente de una población (con o sin substitución), se selecciona el mejor individuo de este grupo para el procesamiento, y se repite hasta que la población está completa. Los torneos se dan frecuentemente entre pares de individuos (torneo tamaño s=2).

Genitor. Este método trabaja individuo por individuo, eligiendo un descendiente de acuerdo al ordenamiento lineal y eligiendo el peor individuo de la población presente para substituirlo.

II.2.5.2. Operadores Genéticos.

Los operadores genéticos pueden ser divididos en dos categorías principales:

Recombinación. Estos operadores generan el intercambio de información entre pares de individuos o grupos de éstos (ej. operador de cruzamiento).

Mutación. Este operador ocasiona el cambio de estado de una localidad o bit de una sola estructura de acuerdo a alguna regla probabilística.

Operador de Cruzamiento.

El operador de cruzamiento (crossover), es un proceso que genera una recombinación de los alelos mediante el intercambio de segmentos entre pares de cromosomas (intercambio de nociones para llegar a formar ideas) [10][14].

El operador genético de cruzamiento es un operador aleatorio de intercambio estructurado. Este operador puede ejecutarse en dos pasos: primero, dos cadenas individuales son seleccionadas por el esquema de reproducción; segundo,estas cadenas pueden ser combinadas de varias maneras:

En el **cruzamiento en un punto** una posición **k** a lo largo de la cadena es seleccionada aleatoriamente entre el rango de uno y la longitud de la cadena menos uno **[1, l-1]**. A partir de ésto, dos nuevas cadenas son creadas al intercambiar todos los caracteres entre las posiciones **k+1** y **l**. Por ejemplo, se tienen dos cadenas de longitud 5 definidas como A=11111 y B=00000; si en la selección aleatoria para la localización de cruce se tiene $k=3$, se obtendrán dos nuevas cadenas definidas como A'=**11100** y B'=**00011**, y dichas cadenas constituirán dos individuos de

la nueva población (Figura 2.2.3.a.)

El **cruzamiento en dos puntos** trata a la cadena (cromosoma) como un anillo. Dos únicos puntos son seleccionados aleatoriamente, rompiendo el anillo en dos segmentos que son intercambiados entre los padres para producir dos descendientes.

El **cruzamiento multipunto** es una extensión del anterior. Así como en el cruzamiento en dos puntos, trata el cromosoma (cadena) como un anillo cuyos puntos de cruzamiento van a cortarlo en segmentos. Los segmentos de los descendientes son alternados entre los dos padres, y se presentan tantos segmentos como puntos de cruzamiento se tengan (figura 2.2.3.b.). Alternativamente, si hay un número impar N de puntos de cruzamiento, un punto puede ser ignorado, o el punto final de la cadena debe ser manejado como el N+1 punto de cruzamiento.

El **cruzamiento segmentado** es una variante del cruzamiento multipunto el cual permite que el número de puntos de cruzamiento sea variable. Esto parte de una aproximación del cruzamiento multipunto a una distribución binomial con una probabilidad de 0.5, y para ésto, el número de puntos de cruzamiento necesita ser variable con una media de L/2, en donde L es la longitud de la estructura. En lugar de tener un número fijo de puntos de cruzamiento, se especifica un rango de segmentos. Este rango especifica la probabilidad de que un segmento termine en un punto de la cadena. Por ejemplo, si el rango es de 0.25, entonces un segmento se inicia presentando una probabilidad de 0.25 en cada bit de la cadena de que éste termine y se inicie un nuevo segmento, cortando la estructura de la cadena. Con un rango de 0.25 el número esperado de segmentos va a ser de L/4. Sin embargo, el cruzamiento en L/4 puntos es diferente al segmentado ya que éste último presentará puntos de cruzamiento variables.

El **cruzamiento uniforme** intercambia bits en lugar de segmentos. Para cada una de las posiciones de los bits en la cadena, los bits de los dos padres son intercambiados con probabilidad P fija. Algo importante que presenta este tipo de cruzamiento es el hecho de que la probabilidad de intercambio de dos bits en una cierta posición de la cadena, es independiente de la selección hecha con respecto a alguna otra posición. El número esperado x de bits intercambiados es $p*L$, y x va a presentar una distribución binomial.

El **cruzamiento mezclado** es similar al cruzamiento en un punto a excepción de que aleatoriamente mezcla las posiciones de los bits de las dos cadenas antes de cruzarlas; después los segmentos de la derecha del punto de cruzamiento son intercambiadas. La operación de mezcla es independiente del número de puntos de cruzamiento. Este método de cruzamiento puede combinarse con el multipunto y el segmentado como también, el cruzamiento simple (tradicional).

De acuerdo a lo descrito, el poder que los algoritmos genéticos presentan está dado básicamente por el operador de reproducción más la acción del operador de cruzamiento:

GA = Reproducción + Cruzamiento

Los algoritmos genéticos combinan nociones (*subcadenas*) para formar nuevas ideas (nuevas *cadenas*). La combinación de nociones es el proceso que el operador de cruzamiento ejecuta.

Sin embargo, el operador de mutación también es necesario e importante, aunque la reproducción y el cruzamiento buscan y recombinan de manera efectiva las estructuras existentes, ocasionalmente pueden perder material genético importante (1's ó 0's en localidades específicas). En los sistemas genéticos artificiales, el operador de mutación es una protección contra pérdidas prematuras de dicho material genético.

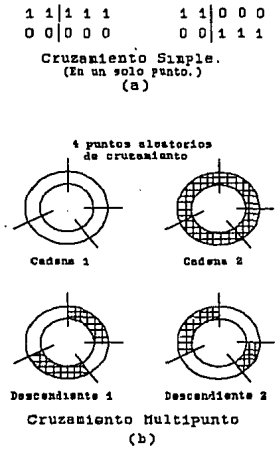


Figura 2.2.3. Operador Genético de Cruzamiento.

Operador de Mutación.

La mutación genética es un proceso en donde un alelo de un gen es aleatoriamente reemplazado por otro para producir una nueva estructura. Generalmente, la probabilidad de mutación en cada gen en la estructura es pequeña; y además, la posibilidad de mutación de cada posición es independiente de la acción en otra posición.

En forma general, los Algoritmos Genéticos trabajan con un código binario, aunque no están limitados a ello. El operador de mutación altera ocasionalmente (baja probabilidad) la posición de un bit; ésto es, el cambio de 1 por 0 ó viceversa.

La probabilidad de mutación es generalmente muy pequeña y puede ser implementada de múltiples formas [12]:

1. Puede darse una probabilidad de mutación constante a través de todas las generaciones (ver figura 2.2.4.a.).
2. Como las cadenas aleatorias corresponden a la población inicial, éstas pueden ser generadas por medio del operador de mutación, mutando los bits de alguna cadena con una probabilidad de 0.5, dando una distribución de probabilidades como que se muestra en la figura 2.2.4.b.
3. Otra forma de dar las probabilidades de mutación es, dada a través de una función de probabilidad exponencialmente decreciente al paso de las generaciones. (Figura 2.2.4.c.).
4. Otra manera de implementar la probabilidad de mutación es mediante una exponencial creciente en cada generación. Esto es, cada que inicia el proceso de una nueva generación, la probabilidad de mutación es pequeña; pero conforme va avanzando, esta probabilidad aumentan hasta que se inicia la generación siguiente que vuelve a partir de una baja probabilidad de mutación. (Figura 2.2.4.d.).

Algo muy importante referente al operador de mutación es que, cuando se tiene una probabilidad de mutación muy alta se presentan mayores riesgos de alteración del material genético ocasionando algunas veces pérdidas de información.

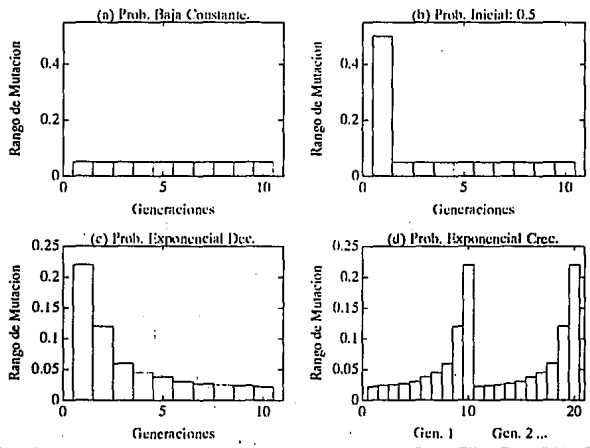


Figura 2.2.4. Operador de Mutación.

II.2.6. TEOREMA DEL ESQUEMA.

Un análisis matemático formal del comportamiento de un Algoritmo Genético, puede llevarse a cabo observando las similitudes que existen en una población de cadenas y su valor de fitness para guiar u orientar la búsqueda hacia una mejor solución.

El esquema (Holland, 1968-1975), es un modelo de similitudes que describe un subconjunto de cadenas con semejanzas en ciertas posiciones de las mismas [14][19]. Como se mencionó anteriormente, los cromosomas (cadenas) de una población están formadas por elementos que pertenecen a un alfabeto binario. Los esquemas manejan un elemento más en este alfabeto que es el carácter * (no importa), formando cadenas a partir de este alfabeto terciario {0,1,*}. Un esquema se relaciona con una cadena en particular si en cada una de las posiciones se tiene un 1 relacionado con un 1, un 0 con un 0 y un * con cualquier valor (1 ó 0). Ejemplo: el esquema **010 describe un subconjunto formado por 4 cadenas {00010, 01010, 10010, 11010}.

La finalidad de los esquemas es proporcionar una forma un tanto sencilla de obtener las similitudes de las cadenas que forman una cierta población a partir de un alfabeto finito, obteniéndose una mayor información y guiándonos hacia una mejor búsqueda.

El número total de posibles esquemas está dado por $(k+1)^l$, siendo k el número de caracteres que forman el alfabeto y l la longitud de las cadenas. Este teorema emplea alfabetos pequeños, ya que maximizan el número de esquemas por bit. Para obtener el límite del número de esquemas de una población, primero se obtiene el número de esquemas posibles en una cadena individual, obteniéndose un límite superior del total de esquemas en la población. En general, una cadena contiene 2^l esquemas distintos. Una población de tamaño n tiene entre 2^l y $n*2^l$ esquemas, dependiendo de la diversidad de la población.

Los esquemas de selección y los operadores genéticos básicos descritos juegan un papel importante en el número de esquemas procesados por los AG's. El operador de reproducción da a las cadenas con mayor adaptabilidad una mayor probabilidad de selección, incrementándose el número de muestras del patrón que presenta más similitudes. El operador de cruzamiento deja sin cambios el esquema si este no lo divide, pero puede elimilarlo si el cruzamiento lo divide. Los esquemas de longitud más corta quedan sin cambios (por el cruzamiento) y se reproducen varias muestras (por la reproducción). La mutación, en rangos bajos, no rompe un esquema frecuentemente. Por la alta compatibilidad de los esquemas de longitud corta (que se llamarán bloques constructores), se propagan de generación en generación dando un incremento exponencial de las mejores cadenas, realizándose en paralelo con la población de n cadenas (paralelismo implícito).

Con objeto de analizar el **Teorema del Esquema** se tomará la siguiente notación:

Se asigna a una cadena de una población con letra mayúscula la cual toma valores del alfabeto V={0,1}, y a cada carácter individual que la forma, con letras minúsculas: $A=a_1a_2a_3a_4a_5a_6a_7$. Cada una de las a_i representa un solo carácter binario (a_i's genes), donde cada carácter puede tomar un 1 ó 0 (alelos, valores de los a_i's).

Las búsquedas genéticas requieren de una población de cadenas individuales A_j, j=1, 2, ..., n, contenidas en la población $A(t)$ en el tiempo (o generación) t.

Un esquema se va a representar como H, el cual toma valores del alfabeto V+={0,1,*}.

II.2.6.1. Propiedades de los Esquemas.

Los esquemas presentan dos propiedades importantes: *orden y longitud*. El orden de un esquema H, denotado por $o(H)$, es el número de posiciones con valores de 1 o 0. El esquema 010**1* es de orden 4. La longitud de un esquema H, denotada por $\delta(H)$, es la distancia entre la primera y la última posición de la cadena con un valor específico de 0 ó 1. Por ejemplo, el esquema 010**1* es de orden 5 (δ(H)=6-1=5).

II.2.6.2. Efectos de los Esquemas de Reproducción y Operadores Genéticos en el Comportamiento de los Algoritmos Genéticos.

Esquema de Reproducción.

Para determinar el efecto de la reproducción en el número esperado de esquemas en una población, se parte de un tiempo t dado con m ejemplos de esquemas en la población A(t) donde $m=m(H,t)$.

Durante la reproducción una cadena A_i es seleccionada con probabilidad $p_i=f_i/\Sigma_{pop}f$. Con esto se espera tener $m(H,t+1)$ representaciones del esquema H en la población en el tiempo $t+1$ dada por la ecuación:

$$m(H,t+1)=m(H,t)*n*\frac{f(H)}{\sum_{pop}f} \qquad (2.2.2)$$

donde f(H) es el valor promedio de los valores de la cadenas que representan el esquema H en el tiempo t. Como $f=\Sigma_{pop}f/n$:

Resumiendo, todo esquema en una población crece o decae de acuerdo a sus esquemas promediados bajo la operación de reproducción. Los esquemas arriba del promedio continuan,

$$m(H,t+1)=m(H,t)*\frac{f(H)}{\bar{f}} \qquad (2.2.3)$$

los que se encuentran abajo del promedio decaen.

Operador de Recombinación.

El cruzamiento es una estructura de intercambio de información entre cadenas. Para ver cuales esquemas son efectados por el operador de cruzamiento y cuales no, consideramos:

A = 0 1 1 1 0 0 0
H_1 = * 1 * * * * 0
H_2 = * * * 1 0 * *

H_1 y H_2 están representados en A. Para ver el efecto del cruzamiento en estos dos esquemas, se va a definir un punto de cruce en la posición 3:

A = 0 1 1|1 0 0 0
H_1 = * 1 *|* * * 0
H_2 = * * *|1 0 * *

El esquema H_1 será destruido porque el 1 en la posición 2 y el 0 en la posición 7 serán colocados en diferentes descendientes. El esquema H_2 se va a seguir manteniendo por el 1 en la posición 4 y el 0 en la posición 5 serán transmitidos a un solo descendiente. El esquema H_1 presenta menos probabilidad de sobrevivencia que el esquema H_2 porque en promedio el punto de cruce presenta mayor probabilidad de caer entre las posiciones extremas. Cuantificando, el esquema H_1 tiene una longitud de 5. Si la localización del cruzamiento se selecciona uniforme y aleatoriamente en las posibles posiciones l-1=7-1=6, el esquema H_1 es destruido con probabilidad $P_d=\delta(H_1)/(l-1)=5/6$. Para el esquema H_2, $\delta(H_2)=1$, y $P_d=1/6$.

Un esquema sobrevive cuando la localización del cruce cae fuera de la longitud definida, la probabilidad de sobrevivencia bajo el cruzamiento es $P_s=1-\delta(H)/(l-1)$. Como el cruzamiento es una elección aleatoria del cruce, se tiene que:

$$P_s \geq 1-P_c*\frac{\delta(H)}{l-1} \qquad (2.2.4)$$

Al considerar la combinación del efecto de los operadores de reproducción y cruzamiento, se obtiene:

$$m(H,t+1) \geq m(H,t) * \frac{f(H)}{\bar{f}} [1 - P_c * \frac{\delta(H)}{l-1}] \qquad (2.2.5)$$

El efecto combinado del cruzamiento y la reproducción es obtenido multiplicando el número esperado de esquemas para la reproducción por la probabilidad de sobrevivencia bajo el cruzamiento P_s.

Operador de Mutación.

El operador de mutación se aplica a cada una de las posiciones de una cadena independiente con probabilidad de sobrevivencia por la influencia de este operador de $(1-P_m)$. Un esquema en particular sobrevive cuando cada una de las posiciones de $o(H)$ en el esquema sobreviven. La probabilidad de sobrevivencia por este operador se obtiene multiplicando $(1-P_m)$, $o(H)$ veces $((1-P_m)^{o(H)})$. Para valores de P_m pequeños ($P_m << 1$), la probabilidad de sobrevivencia de un esquema se aproxima a $1-o(H)*P_m$. A partir se ésto se llega a la expresión que representa el número esperado de copias que va a recibir la generación siguiente de un esquema H en particular bajo la influencia de los tres operadores genéticos básicos (reproducción, cruzamiento y mutación):

$$m(H,t+1) \geq m(H,t) * \frac{f(H)}{\bar{f}} [1 - P_c * \frac{\delta(H)}{l-1} - P_m o(H)] \qquad (2.2.6)$$

donde:

f : Resultado de la evaluación de la función.
H : Esquema.
m : Número de esquemas.
δ : Definición de longitud.
o : Orden.
P_c : Probabilidad de cruzamiento.
P_m : Probabilidad de mutación.
l : Longitud de la cadena.

La expresión anterior indica que esquemas con longitudes pequeñas, orden bajo y con valores arriba del promedio de la población, reciben un número de elementos en generaciones posteriores que se incrementa en forma exponencial. Se ha estimado que un Algoritmo Genético procesa un orden de n^3 esquemas en cada generación aun cuando solamente n elementos son realmente manipulados. Esta característica ha sido denominada como ***"Paralelismo Implícito"*** [33]. La conclusión anterior es conocida como el **Teorema del Esquema, o Teorema Fundamental de los Algoritmos Genéticos.**

II.3. PROCESAMIENTO PARALELO.

La investigación dentro del campo de *Procesamiento Paralelo* es en la actualidad una actividad muy intensa. Esto motivado en gran manera por los recientes avances tecnológicos, que abren la posibilidad de resolver problemas que en el pasado no eran considerados debido a su gran demanda computacional.

Un ejemplo de esto son las aplicaciones en el campo de procesamiento de señales como la descrita en esta tesis, la cual presenta una gran demanda computacional debido a sus requerimientos de alta resolución en tiempo-real. El uso de procesamiento paralelo presenta una respuesta viable a dichos requerimientos.

II.3.1. DEFINICION.

En términos generales, se puede definir al ***procesamiento paralelo*** como la actividad de varias entidades (idénticas o heterogéneas), trabajando conjuntamente hacia un objetivo común [7]. Para el caso del cómputo paralelo, dichas entidades corresponden a computadoras o procesadores.

II.3.2. ARQUITECTURAS DE COMPUTADORAS DE PROCESAMIENTO PARALELO.

Un sistema general de procesamiento paralelo está compuesto de varios elementos de procesamiento (PE's), los cuales pueden operar en paralelo, comunicándose unos con otros cuando sea necesario (Figura 2.3.1). Las architecturas en paralelo difieren con respecto al poder de procesamiento de sus PE's y el grado de interconectividad entre ellos.

Un concepto empleado para caracterizar el poder de los elementos de procesamiento es el concepto de *granularidad*, el cual es una medida del tamaño de las tareas que pueden ser efectivamente ejecutadas por los PE's de una arquitectura específica. Una arquitectura de **grano fino** tiene funcionalidad limitada y un amplio ancho de banda para la comunicación de datos locales. Por el contario, una de **grano mediano** es de propósito general y el ancho de banda de la comunicación entre procesos es reducida.

Otro concepto importante es la *conectividad*, la cual es dependiente del tipo de memoria del sistema. Si todos los procesadores tienen acceso a todos los bancos de memoria, se dice que se tiene una arquitectura de **memoria compartida**, mientras que si cada procesador únicamente tiene acceso a su propia memoria local, se presenta una arquitectura de **memoria distribuída.**

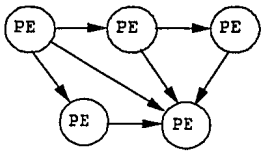


Figura 2.3.1. Sistema General de Procesamiento Paralelo.

La topología de la arquitectura del sistema paralelo nos indica la interconexión directa que cada procesador tiene con los restantes. En la figura 2.3.2 se presentan variaciones en la conectividad de arquitecturas de memoria compartida, mientras que en la figura 2.3.3 se ilustran cinco clases de conectividad para arquitecturas de memoria distribuída.

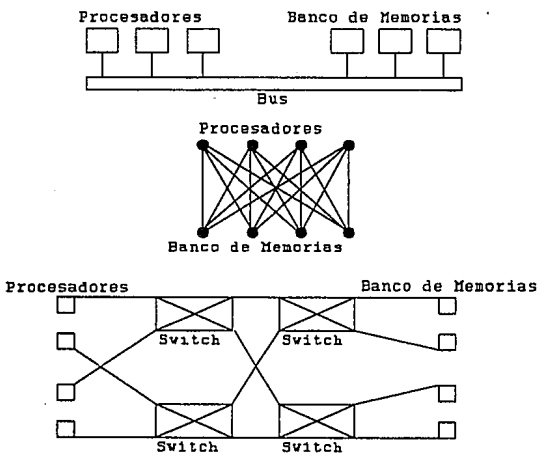
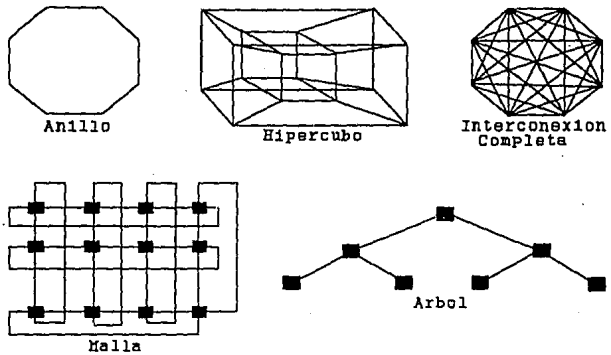


Figura 2.3.2. Arquitecturas de Memoria Compartida.



Arquitecturas de Memoria Distribuida.

Figura 2.3.3. Arquitecturas de Memoria Distribuída.

Clasificación.

Los lenguajes de programación convencionales operan de manera secuencial, es decir, las instrucciones de un programa son ejecutadas una a la vez. Esta naturaleza secuencial se presenta debido a la arquitectura secuencial de las computadoras convencionales (basadas en el concepto de Von Neumann [22]), las cuales presentan un solo procesador, memoria y dispositivos de E/S ligados vía un bus de datos (Figura 2.3.4). Sin embargo, muchos problemas tales como simulaciones, procesamiento de señales, análisis de elementos finitos y control digital entre otras, presentan un paralelismo inherente que puede ser convenientemente explotado. Para tal efecto, los algoritmos para el cálculo de las soluciones de dichos problemas necesitarían ser establecidas en términos paralelos.

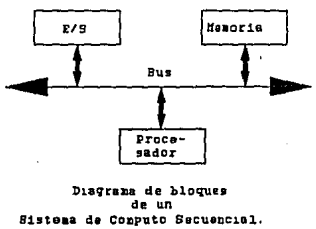


Figura 2.3.4. Sistema de Cómputo Secuencial.

Las arquitectura en paralelo pueden clasificarse de acuerdo a diferentes criterios. Uno de los sistemas de clasificación más ampliamente usados fue introducido por **Flynn** [22][8], quien consideró el modelo secuencial tradicional de Von Neumann como una organización de secuencia simple de instrucciones-secuencia simple de datos (**SISD**).

De acuerdo a la taxonomía de Flynn (Figura 2.3.5), las arquitecturas en paralelo son:

- (**SISD**). Esta arquitectura representa a la mayoría de las computadoras secuenciales. Las instrucciones se ejecutan secuencialmente pero pueden estar sobrepuestas en las etapas de ejecución (segmentación encauzada). Una computadora SISD puede tener más de una unidad funcional.

- (**SIMD**). Esta clase corresponde a los procesadores matriciales. En este tipo de arquitectura existen múltiples elementos de procesamiento (PE's) supervisados por la misma unidad de control. Todos los PE's reciben la misma instrucción emitida por la unidad de control pero operan sobre diferentes conjuntos de datos procedentes de flujos distintos.

- (**MISD**). En esta arquitectura se tienen varios procesadores ejecutando diferentes instrucciones que operan sobre una misma secuencia de datos.

- (**MIMD**). Esta arquitectura consiste de varios procesadores independientes, cada uno capaz de ejecutar diferentes instrucciones sobre diferentes datos. Una variedad de topologías existen para las interconexiones (Figura 2.3.6). Como ejemplo de arquitectura MIMD se tiene al **Transputer**.

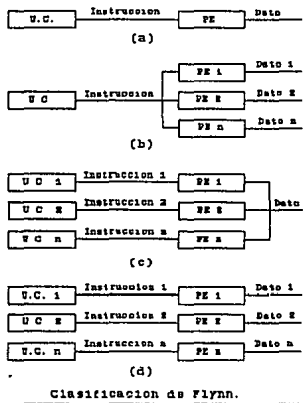
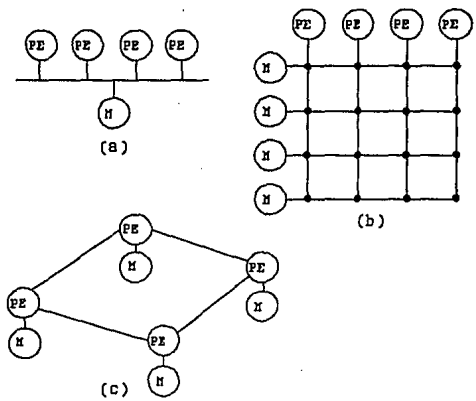


Figura 2.3.5. Taxonomía de Flynn.



Configuracion Memoria-Procesador MIMD.
(a) Bus Compartido; (b) Crossbar; (c) Punto a Punto.
PE=Elemento de Procesamiento; M=Memoria.

Figura 2.3.6. Configuración Memoria-Procesador MIMD.

No obstante que la Taxonomía de Flynn es ampliamente utilizada y provee un marco inicial de clasificación de sistemas de procesamiento paralelo, existen aspectos en éstos que no son tomados en cuenta por dicha taxonomía. En particular no distingue el grado de "acoplamiento" que existe en un sistema como resultado de compartir recursos como: buses, memoria, sistemas de reloj, etc.

Evaluación del Desempeño de un Sistema de Procesamiento Paralelo.

Las métricas más comúnmente empleadas para medir el desempeño de un sistema paralelo son el *tiempo de ejecución, costo/desempeño, speed-up, eficiencia y fracción serial.*

Para determinar que tan eficientemente se está usando un procesador paralelo, el speed-up y la eficiencia representan un buen patrón métrico. El speed-up es generalmente medido al correr el mismo programa en un número determinado de procesadores. La relación de speed-up es el tiempo tomado por un solo procesador dividido entre el tiempo tomado por p procesadores al ejecutar la misma tarea. Esto es:

$$s = \frac{T[1]}{T[p]} \qquad (2.3.1)$$

Escalando la relación de speed-up por el número de procesadores, se obtiene una medición más representativa del paralelismo. Esta métrica es conocida como *eficiencia (e)* y está definida de la siguiente manera:

$$e = \frac{T[1]}{p * T[p]} = \frac{s}{p} \qquad (2.3.2)$$

Al igual que el speed-up, la eficiencia es una medida lineal y directamente proporcional al paralelismo, con lo que, para valores bajos de eficiencia se dice que los recursos están siendo desaprovechados.

La fracción serial, basada en la forma más simple de la *Ley de Amdahl* [11], es una nueva métrica la cual está definida mediante la siguiente ecuación:

$$f = \frac{1/s - 1/p}{1 - 1/p} \qquad (2.3.3)$$

donde s es el speed-up del sistema formado por p procesadores. Esta nueva medida del desempeño de sistemas paralelos proporciona una mayor información sobre la distribución de cargas (tareas) y la comunicación y sincronización de los procesadores que integran el sistema.

II.3.3. El TRANSPUTER Y OCCAM.

El Transputer.

El **Transputer** (contracción de las palabras: *transistor* y *computer*) pertenece a una nueva generación de arquitecturas VLSI, la cual soporta concurrencia y sincronización explícita [25].

El transputer es una familia de microprocesadores con diferentes capacidades. El IMS T800 consiste de un procesador de 32 bits, una unidad de punto flotante de 64 bits, 4 Kbytes de memoria, 4 enlaces seriales de comunicación de alta velocidad (*links*) y una interfaz de memoria externa. La arquitectura del transputer [25] se presenta en la figura 2.3.7.

El microprocesador corre a 20 MHz pudiendo entregar 10 MIPS. Su set de instrucciones y sus 32 registros están diseñados para maximizar su velocidad de ejecución. Además presenta

un programa de prioridad el cual permite el tiempo compartido entre un número de procesos pudiéndose ejecutar dichos procesos en modo secuencial convencional o concurrentemente[1]

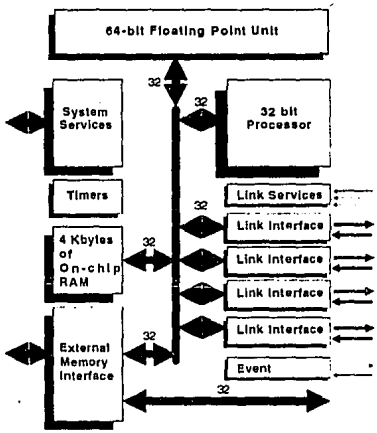


Figura 2.3.7. Arquitectura del Transputer T800.

Los links son bidireccionales permitendo las conexiones punto a punto entre transputers o entre un transputer y dispositivos externos. En un solo link se implementan dos canales, uno en cada dirección. Los datos son enviados en forma serial en base a un determinado protocolo para proporcionar automáticamente la sincronización, presentando una transferencia de datos segura.

OCCAM.

OCCAM es un lenguaje de alto nivel especialmente diseñado para ser utilizado eficientemente en sistemas de **Transputers** [8][23][24].

[1] El procesamiento concurrente es la ejecución de procesos en "paralelo" en un mismo procesador, mientras que el procesamiento paralelo es la ejecución de procesos en paralelo en procesadores separados.

El diseño del lenguaje **OCCAM** fue originalmente influenciado por el modelo teórico **CSP (Communicating Sequential Process)** introducido por Hoare [18]. Los principios para el diseño de OCCAM se derivaron del pensamiento del filósofo del siglo XIV William de Ockham. Dicho principio, denominado "**Occam's Razor**", establece que "todos los fenómenos observables deben ser explicados en su forma más simple".

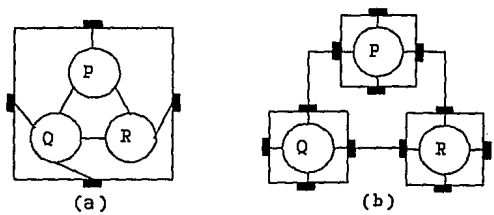
La unidad básica de programación en Occam es un *proceso.* Cada proceso puede ser visto como una caja negra, el cual puede comunicarse con otros procesos usando *cánales de comunicación* punto a punto.

Todos los procesos en Occam son finitos, ya que inician, realizan un número de acciones y terminan. Una acción puede ser un conjunto de procesos secuenciales ejecutados uno después del otro, tal como ocurre en un lenguaje de programación convencional, o un conjunto de procesos en paralelo ejecutados todos al mismo tiempo. Ya que un proceso puede estar compuesto a su vez de procesos, éste puede presentar concurrencia interna.

Occam no permite el uso de variables compartidas. En lugar de ésto, el dato compartido es comunicado a los procesos a través de canales de comunicación. Un punto importante para la comunicación es que ésta es sincronizada, es decir, si un canal es usado como entrada en algún proceso y como salida en otro, la comunicación va a tomar lugar cuando ambos procesos estén listos para enviar y recibir.

Otra característica de Occam es que puede ser usado para programar un solo transputer o una red de los mismos. Cuando Occam se usa para programar una red de transputers, cada transputer ejecuta el proceso localizado en él, la comunicación entre procesos colocados en transputers diferentes es implementada directamente a través de los links del transputer.

En la figura 2.3.8.a. se presentan tres procesos corriendo en un solo transputer y comunicándose entre sí, y en la figura 2.3.8.b. se presentan los tres procesos pero esta vez implementado cada proceso en un transputer diferente comunicándose a través de los links del transputer [25].



Procesos Mapeados en Uno o Varios Transputers.

Figura 2.3.8. Comunicación de Procesos.

Primitivas de Occam.

Como se mencionó anteriormente, el principio de OCCAM es mantener el lenguaje en forma más simple. De esta misma forma los procesos en OCCAM pueden construirse a partir de tres primitivas:

* **Asignación** v:=e
  donde a la variable 'v' se le asigna el valor de la expresión 'e'.

* **Entrada** c ? e
  donde un valor es solicitado del canal 'c' y almacenado en la variable 'v'.

* **Salida** c ! e
  donde el valor de la expresión 'e' sale al canal 'c'.

OCCAM provee otros dos procesos primitivos de mucha utilidad **SKIP** y **STOP. SKIP** es un proceso que inicia, no efectua acción alguna y termina, mientras que STOP es un proceso que inicia, no efectua acción alguna y nunca termina.

Constructores de Occam.

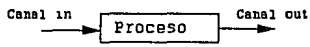
Varios procesos primitivos pueden combinarse en un proceso más grande para formar un constructor, el cual es por si mismo un proceso y puede usarse como componente de otro constructor. Estos constructores son:

* **SEQ** constructor secuencial.
* **PAR** constructor paralelo.
* **ALT** constructor de alternación.

SEQ significa que las instrucciones en un proceso van a ser ejecutadas una tras otra. En el siguiente código se ejemplifica el uso de este constructor:

```
SEQ                          SEQ
  factor:=1.2                  ... proceso 1
  canalin ? x                  ... proceso 2
  nuevo:=factor*x              ... proceso 3
  canalout ! nuevo
```

La sangría es usada para indicar la estructura del programa, denotándo que las cuatro operaciones después de SEQ son parte de ese proceso o, en el otro caso, que los tres proceso después de SEQ son parte de otro proceso mayor. En la figura 2.3.9 se ilustra la relación entre proceso y canales.



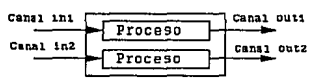
Constructor SEQ.

Figura 2.3.9. Constructor SEQ.

PAR nos indica que los procesos seguidos de este constructor serán realizados en paralelo:

```
PAR
  ... proceso 1
  ... proceso 2
```

En general, los procesos paralelos corren asíncronamente y la sincronización solo es necesaria cuando los procesos necesitan comunicarse a través de un canal. En la figura 2.3.10 se presentan dos procesos ejecutados en paralelo.



Constructor PAR.

Figura 2.3.10. Constructor PAR.

ALT es usado cuando un subconjunto de canales de entrada pueden ser usados para inicializar cálculos en un procesos. ALT opera un procedimiento de "primero en ganar", en el cual solo el proceso asociado con la primera entrada será ejecutado. En el siguiente ejemplo, si la entrada 2 es la primera en producir una entrada, entonces solo el proceso 2 será ejecutado.

```
ALT
  entrada1 ? x
    ... proceso 1
  entrada2 ? x
    ... proceso 2
  entrada3 ? x
    ... proceso 3
```

Así mismo, algunas construcciones más convencionales se incluyen en OCCAM, como son: **IF, CASE, WHILE.**

Prioridades.

Frecuentemente en las aplicaciones en tiempo-real algunos procesos requieren más atención que otros. Esto es una característica de OCCAM que permite a procesos en construcciones **PAR** y **ALT** asignarles una prioridad de ejecución, es decir, procesos de alta prioridad son seleccionados preferentemente que los procesos de baja prioridad.

PRI PAR. La construcción de prioridad PAR provee un mecanismo para especificar la prioridad en la cual los procesos componentes se ejecutarán. El formato de una prioridad PAR es:

```
PRI PAR
  proceso 1
  .
  .
  .
  procesos n
```

donde el proceso 1 es el de prioridad más alta y los otros procesos componentes tienen una prioridad en forma decreciente de acuerdo a su orden textual. También, un número de procesos pueden compartir el nivel de prioridad al ser agrupados en un constructor PAR.

```
PRI PAR
  PAR
    p1
    p2
  PAR
    p3
    p4
```

PRI ALT. Cuando se tienen varias entradas listas al mismo tiempo, el proceso asociado con la entrada de más alta prioridad es el seleccionado para su ejecución.

```
PRI ALT
  entrada 1
    proceso 1
  .
  .
  .
  entrada n
    proceso n
```

II.4. ESTIMACION ESPECTRAL.

Muchos de los fenómenos que ocurren en la naturaleza se caracterizan mejor estadísticamente en términos de resultados y valores promedios.

Debido a las variaciones aleatorias de muchas señales, se ha optado por un punto de vista estadístico para el análisis de las mismas, el cual trate con las características promedio de estas señales. En estas situaciones la Estadística juega un papel importante en conjunto con las técnicas de estimación de señales para permitir la extracción de información valiosa de la señal bajo análisis.

El Análisis Espectral es una herramienta utilizada para caracterizar el contenido de frecuencia de una señal, comúnmente realizada por medio de la Transformada de Fourier, siendo los fundamentos matemáticos para relacionar señales en el tiempo con su representación en el dominio de la frecuencia.

Los métodos de estimación espectral basados en el cálculo de la Transformada de Fourier de una señal son usualmente referidos como métodos de estimación convencionales [26]. En las últimas décadas, se han desarrollado estimadores espectrales, con la finalidad de tener una mayor resolución. Muchos de estos métodos "modernos" han fincado sus raíces en el campo del análisis de series de tiempo y la teoría de aproximación.

Con el rápido incremento en las capacidades de cómputo, ha sido posible llevar estos métodos teóricos a la práctica.

II.4.1. SEÑALES ALEATORIAS.

Para una señal aleatoria, el valor de la forma de onda o el valor de la señal no está especificado en cada instante de tiempo, es decir, no es posible predecir su futuro con certeza tomando como base lo que se conoce de la misma.

Una señal puede ser aleatoria en una variedad de formas. El tipo más común es la que presenta una magnitud aleatoria, figura 2.4.1(a). Así mismo, existen otro tipo de señales, figura 2.4.1(b), que pueden tener solo dos niveles ('1' o '0'); pero la transición entre estos dos niveles ocurre en instantes aleatorios. En la figura 2.4.1(c) se presenta los dos tipos de señales aleatorias mencionadas, tanto en magnitud como en ocurrencia.

Usualmente un método utilizado para caracterizar las señales aleatorias es determinar alguna de las propiedades promedio de la señal [27].

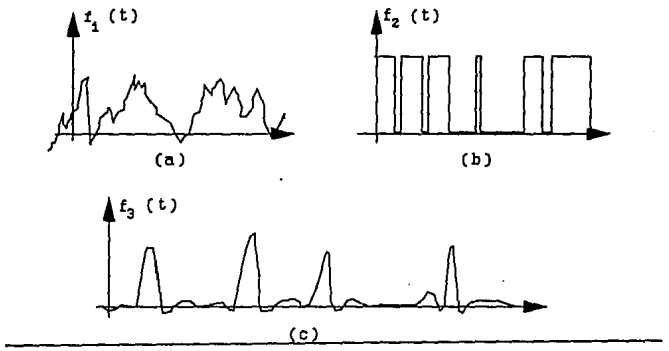


Figura 2.4.1. Señales Aleatorias.

II.4.2. METODOS DE ESTIMACION ESPECTRAL.

El problema general de la estimación espectral es la determinación del contenido espectral de un proceso aleatorio basado en un conjunto finito de observaciones de ese proceso. Formalmente, la ***Densidad de Potencia Espectral (PSD)*** [26][28], la cual se denota por $\mathbf{P_{xx}(f)}$, de un proceso aleatorio ***x[n]*** es definida como:

$$P_{xx}(f) = \sum_{k=-\infty}^{\infty} r_{xx}[k]\exp(-j2\pi fk) \qquad (2.4.1)$$

Estimación Espectral Tradicional.

Los métodos de Estimación Espectral tradicionales se basan en el análisis de Fourier. Los principales métodos son el **Periodograma**, originalmente propuesto por Schuster en 1898 [26][28] y el estimador espectral de **Blackman-Tukey** [26][28].

Peridiograma.

Este método de estimación espectral es inconsistente, ya que ni siquiera el valor promedio converge al valor real aunque la longitud del segmento de datos se incremente y la variancia es una constante.

Este estimador espectral es definido como:

$$P_{PER}(f) = \frac{1}{N} \left| \sum_{n=0}^{N-1} x[n] \exp(-j2\pi fn) \right|^2 \qquad (2.4.2)$$

donde:

N es el número de datos disponibles.
x[n] son cada uno de los datos.

Si se tiene una frecuencia dada f_o, el periodograma puede reescribirse de la siguiente forma

$$P_{PER}(f_o) = N \left| \sum_{k=0}^{N-1} h[n-k] x[k] \right|^2 \qquad (2.4.3)$$

La variancia , en el periodograma, es una constante independiente de N, y se define como:

$$var[P_{PER}(f)] = P^2_{xx}(f) \qquad (2.4.4)$$

Blackman-Tukey.

El estimador espectral de **Blackman-Tukey** está definido como

$$P_{BT}(f) = \sum_{k=-(N-1)}^{N-1} w[k] r_{xx} \exp(-j2\pi fk) \qquad (2.4.5)$$

donde **w[k]** es una secuencia real llamada una **ventana de retraso** con las siguientes propiedades:

1. $0 \le w[k] \le w[0]=1$
2. $w[-k]=w[k]$
3. $w[k]=0$ para $|k| > M$

siendo **N** el número de datos y **M ≤ N-1**. De acuerdo a la última propiedad de la ventana de retraso, la ecuación anterior puede ser escrita como

$$P_{BT}(f) = \sum_{k=-M}^{M} w[k] r_{xx}[k] \exp(-j2\pi fk) \qquad (2.4.6)$$

y es conocida como **estimador espectral de Blackman-Tukey (BT)**.

La variancia de este estimador está definida por:

$$var[P_{BT}(f)] = \frac{P^2_{xx}(f)}{N} \sum_{k=-M}^{M} w^2[k] \qquad (2.4.7)$$

Estimación Espectral Paramétrica.

Muchos procesos aleatorios en tiempo discreto que se encuentran en la práctica son aproximados por un modelo de ***series de tiempo o función de transferencia racional.*** En este modelo una secuencia de entrada **u[n]** y una secuencia de salida **x[n]** están relacionadas por la siguiente ecuación lineal [26][28]:

$$x[n] = -\sum_{k=1}^{p} a[k]x[n-k] + \sum_{k=0}^{q} b[k]u[n-k] \qquad (2.4.8)$$

Dicha ecuación es conocida como modelo **ARMA (autoregressive-moving average)** [26][28] y se muestra en la figura 2.4.2. Lo interesante en este modelo radica en la relación de los filtros lineales con la función de transferencia racional.

u[n] es una parte innata del modelo y proporciona la naturaleza aleatoria del proceso observado **x[n]**.

La función de transferencia **H(z)** entre la entrada **u[n]** y la salida **x[n]** para el proceso ARMA es la función racional

$$H(z) = \frac{B(z)}{A(z)} \qquad (2.4.9)$$

donde **A(z)** es la transformada **z** de **AR**:

$$\sum_{k=0}^{p} a[k]z^{-k} \qquad (2.4.10)$$

y **B(z)** es la transformada **z** de **MA**:

$$\sum_{k=0}^{q} b[k]z^{-k} \qquad (2.4.11)$$

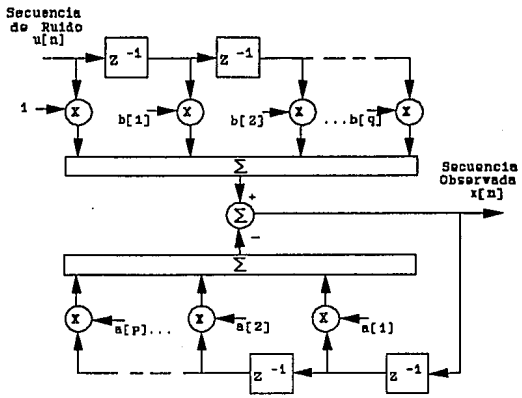


Figura 2.4.2. Modelo ARMA.

Se asume que **A(z)** tiene todos los ceros dentro del círculo unitario del plano-z. Esto garantiza que **H(z)** sea estable y el filtro causal.

La transformada **z** de la función ACF [26] a la salida de un filtro lineal $\mathbf{P_{xx}(z)}$, está relacionada con la entrada $\mathbf{P_{uu}(z)}$, por la siguiente expresión:

$$P_{xx}(z)=H(z)H^*(1/z^*)P_{uu}(z)=\frac{B(z)B^*(1/z^*)}{A(z)A^*(1/z^*)}P_{uu}(z) \qquad (2.4.12)$$

Cuando la ecuación anterior es evaluada a lo largo del círculo unitario, z=exp(j2πf) para $-½ \le f \le ½$, se llega a la **densidad espectral de potencia** $\mathbf{P_{xx}(f)}$. Asumiendo que el proceso de entrada es una secuencia de ruido blanco con media cero y variancia σ^2, la **PSD** del ruido es entonces σ^2. Por tanto, la PSD del proceso de salida ARMA es:

$$P_{ARMA}(f)=P_{xx}(f)=\sigma^2\left|\frac{B(f)}{A(f)}\right|^2 \qquad (2.4.13)$$

La especificación de los parámetros **a[k]** (coeficientes autorregresivos), **b[k]** (coeficientes promedio variable), y σ^2 es equivalente a especificar la **PSD** del proceso **x[n]**.

Si todos los coeficientes a[k] se eliminan (excepto a[0]=1) en el proceso ARMA,

$$x[n]=\sum_{k=0}^{q} b[k]\,u[n-k] \tag{2.4.14}$$

el proceso es estrictamente un proceso **MA (moving average)** de orden **q**. Por tanto:

$$P_{MA}(f)=\sigma^2\,|B(f)|^2 \tag{2.4.15}$$

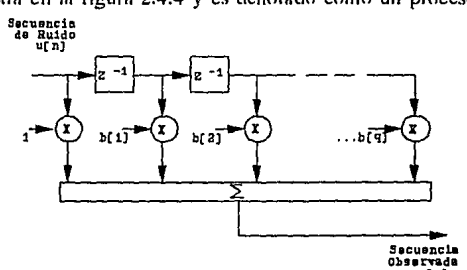
Este modelo se presenta en la figura 2.4.3 y es denotado como un proceso **MA(q)**. Ahora bien, si todos los coeficientes b[k] excepto b[0] son cero en el modelo ARMA,

$$x[n]=-\sum_{k=1}^{p} a[k]\,x[n-k]+u[n] \tag{2.4.16}$$

el proceso es estrictamente un proceso **AR** de orden p. El proceso es llamado de **autorregresión** en el que la secuencia **x[n]** es una regresión lineal de sí misma con **u[n]** representando el error. En este modelo, el valor presente es una suma de valores anteriores más un término de ruido. La PSD es entonces:

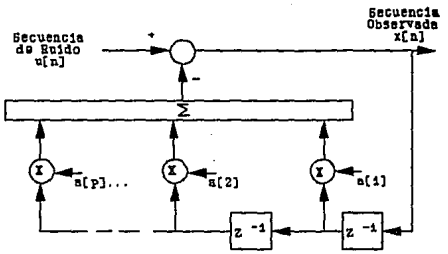
$$P_{AR}(f)=\frac{\sigma^2}{|A(f)|^2} \tag{2.4.17}$$

Este proceso se muestra en la figura 2.4.4 y es denotado como un proceso **AR(p)**.



Modelo MA.

Figura 2.4.3. Modelo AR.



Modelo AR.

Figura 2.4.4. Modelo MA.

II.4.3. CRITERIO DE SELECCION COSTO/BENEFICIO.

En los métodos de estimación espectral paramétrica, la selección del orden del modelo del estimador es una tarea importante. Existen varios criterios para la selección del orden del modelo de un estimador espectral aplicables exclusivamente a métodos paramétricos. Algunos de estos métodos son: el criterio de *Error de Predicción Final (Final Prediction Error, FPE)* [26] aplicable a métodos paramétricos autorregresivos, el *Criterio de Información de Akaike (AIC)* [26] y el criterio de *Función de Transferencia Autorregresiva (CAT)* [28].

El desempeño de estos criterios de selección son similares, siendo los métodos FPE y CAT particulares de los modelos AR, y el criterio de Akaike un método más general.

Esta sección presenta el criterio de *Costo/Beneficio* [31], el cual permite la comparación simultánea de estimadores paramétricos y no paramétricos para evaluar el desempeño de estos métodos.

De manera general, el método de estimación espectral seleccionado de un conjunto de $m=1$ a $m=M$ métodos bajo consideración, será el método que presente el mínimo valor para una función de criterio óptimo $C_o(m)$. Esto es:

$$c_o(m) = \min_{ij}\left\{\sum_{k=1}^{K} w_k,\ c(m,k,i,j),\ i=0,1,...,I,\ j=0,1,...,J\right\} \qquad (2.4.18)$$

donde $k=1,...,K$ representa las k diferentes señales de prueba consideradas; los índices i y j son

los indicadores del orden para los modelos AR y MA, respectivamente, tal que, para *i=j=0* se tiene un método no paramétrico. La utilidad de cada una de las señales aplicadas al criterio puede expresarse con una función de peso w_k.

La función **c(m,k,i,j)** es la función costo/beneficio, representando el numerador el *costo* de aplicar un método de estimación *m* a una cierta señal *k* de *N datos*, siendo el *costo* una medida de la complejidad computacional del algoritmo empleado. El denominador representa el beneficio, siendo éste una medida de la calidad del estimador espectral.

II.5. REFERENCIAS.

[1] Austin, S., "Genetics Solutions to XOR Problems", AI, Vol. 5, Num. 2, pp. 52-57, 1990.

[2] Bagley, J.D., "The Behavior of Adaptive Systems Which Employ Genetic and Correlation Algorithms", Dissertation Abstracts International, University of Michigan, 1967.

[3] Baker, J.E., " Reducing Bias and Inefficiency in the Selection Algorithms", Proceedings of the Second International Conference on Genetic Algorithms, pp. 14-21, 1987.

[4] Booker, L.B., "Intelligent Behavior as an Adaptation to the Task Environment", Dissertation Abstracts International, Ann Arbor: University of Michigan, 1982.

[5] Box, G., "Evolutionary Operation: A Method for Increasing Industrial Productivity", Journal of the Royal Statistical Society, pp. 81-101, 1957.

[6] Brindle, A., " Genetic Algorithms for Function Optimization", Doctoral Dissertation, Edmonton: University of Alberta, Department of Computer Science, 1981.

[7] Carlini, U., Villano, U., "Transputer ans Parallel Architectures. Message-Distributed Systems", Ellis Horwood Limited, 1991.

[8] Carling, A., " Parallel Processing. The Transputer ans Occam", Sigma Press, 1988.

[9] De Jong, K.A., "Genetic Algorithms: A 10 Years Perspective", Proceedings of the First International Conference on Genetic Algorithms, pp. 169-177, 1985.

[10] Eshelman, L.J., Caruana, R.A., Schaffer, J.D., " Biases in the Crossover Landscape", Proceedings of the Third International Conference on Genetic Algorithms, San Mateo, Ca: Morgan Kaufmann Publishers Inc., 1989.

[11] Flatt, P., Karp, A.H., "Measuring Parallel Processor Performance", Communications of the ACM, Vol.33, Num.5, pp. 539-543, Mayo 1990.

[12] Fogarty, T.C., "Varying the Probability of Mutation in the Genetic Algorithms", Proceedings of the Third International Conference on Genetic Algorithms, San Mateo, Ca: Morgan Kaufmann Publishers Inc., 1989.

[13] Fogel, L.J., Owens, A.J., Walsh, M.J., "Artificial Intelligence Through Simulated Evolution", New York: John Wiley, 1966.

[14] Goldberg, D.E., "Genetic Algorithms in Search, Optimization, and Machine Learning", MA: Addison-Wesley, 1989.

[15] Goldberg, D.E., Deb, K., "A Comparative Analysis of Selectionn Schemes Used in Genetic Algorithms", Foundations of Genetic Algorithms, IEEE Computer Society Press., Los Alamitos, CA., 1992.

[16] Goldberg, D.E., Koza, J.R., "Genetic Algorithms and Genetics-based Machine Learning", The Tenth National Conference on Artificial Intelligence, 1992.

[17] Grefenstette, J.J., Baker, J.E., "How Genetic Algorithms Work: A Critical Look at Implicit Parallelism", Proceedings of the Third International Conference on Genetic Algorithms, San Mateo, Ca: Morgan Kaufmann Publishers Inc., 1989.

[18] Hoare, C.A.R., "Communicating Sequential Processes", Prentice Hall, 1985.

[19] Holland, J.J., " Adaptation in Natural and Artificial Systems", Ann Arbor, MI: University of Michigan Press., 1975.

[20] Hollstien, R.B., "Artificial Genetic Adaptation in Computer Control Systems", Dissertation Abstracts International, University of Michigan, 1971.

[21] Hord, M.R., "Parallel Processing Supercomputing in SIMD Architectures", CRC Press., 1990.

[22] Hwang, K., Briggs, F., "Arquitecturas de Computadoras y Procesamiento Paralelo", McGraw-Hill, Inc. U.S.A., 1988.

[23] Inmos Limited, "A Tutorial Introduction to Occam Programming", BSP Professional Books, 1987.

[24] Inmos Limited, "Occam 2. Reference Manual.", Prentice Hall, 1988.

[25] Inmos Limited, "Transputer Overview. The Transputer Databook.", 2ª Edición, 1989.

[26] Kay, S.M., "Modern Spectral Estimation", Prentice Hall, 1988.

[27] Lynn, P.A., "An Introduction to the Analysis and Processing Signal", Macmillan Press., 1982.

[28] Marple, S.L., Kay, S.M., "Spectrum Analysis: A Modern Perspective", Proceedings of the IEEE, Vol 69., 1981.

[29] Proakis, J.G., Manolakis, D.G., "Digital Signal Processing", Mcmillan Publishing, 1992.

[30] Rosenberg, R.S., "Simulation of Genetic Populations with Biochemical Properties", Dissertation Abstracts International, University of Michigan, 1967.

[31] Ruano, M.G., "Investigation of Real-Time Spectral Analysis Techniques for Use with Pulsed Ultrasonic Doppler Blood-Flow Detectors", PhD Thesis, University College of North Wales, Bangor, U.K., 1992.

CAPITULO III

INVESTIGACION DE METODOS DE SELECCION Y EL EFECTO DE DIFERENTES PARAMETROS EN ALGORITMOS GENETICOS

En el capítulo anterior se presentaron los conceptos y las bases teóricas de los Algoritmos Genéticos. Retomando estos conceptos podemos ahora implementar este algoritmo para la solución de algún problema en particular.

Con los operadores genéticos básicos contamos con las herramientas para construir un Algoritmo Genético Simple (AGS) y de esta manera poder analizar el comportamiento que presenta a las variaciones de estos operadores en conjunto con diferentes esquemas de reproducción. Los métodos de selección o esquemas de reproducción que se analizan en este capítulo con el objeto de encontrar los diferentes parámetros que nos proporcionen el mejor desempeño del algoritmo son: el método de la Ruleta, el método de Escalamiento y el método Ranking.

La implementación del algoritmo ha sido realizada en **MATLAB (Matrix Laboratory)** [9], el cual trabaja esencialmente con una sola clase de objeto, una matriz rectangular numérica con posibilidad de manejar elementos complejos.

III.1. INTRODUCCION.

Los Algoritmos Genéticos operan sobre un gran espacio de soluciones, y cada elemento de la población representa una solución factible al problema considerado. La representación más comúnmente usada para los individuos de una población es mediante simples cadenas binarias, es decir, cada una de las variables de decisión es codificada como una cadena binaria y éstas a su vez, son concatenadas para formar un cromosoma o genotipo de la población (para el caso de variables múltiples).

Teniendo definida esta representación, el primer paso en un Algoritmo Genético Simple (AGS) es crear un población inicial. Una de las formas de realizar ésto es crear el número

requerido de individuos usando un generador de números aleatorios con distribución uniforme. Por ejemplo, para generar una población binaria de N individuos cuyos cromosomas sean de tamaño L, se deben producir NxL números aleatorios uniformemente distribuidos del conjunto $\{0,1\}$.

Con la definición de la representación de los individuos de una población podemos ahora definir la estructura de datos adecuada para el algoritmo.

III.2. ESTRUCTURA DE DATOS.

Los Algoritmos Genéticos procesan poblaciones de cadenas, por lo tanto, la estructura de datos básica es una cadena. Una población se va a construir como un arreglo de elementos donde cada uno de ellos debe presentar la siguiente información:

* Fenotipo o variables de decisión.
* Genotipo o cromosoma artificial.
* Valor de la función objetivo.
* Valor de la función de fitness.

Cromosoma. Esta estructura, la cual es una cadena de bits, representa un individuo de la población que además corresponde a una posible solución. Esta estructura está representada en la implementación del algoritmo (Apéndice A) por la variable **crom.**

Fenotipo. Las variables de decisión o fenotipo se obtienen en los algoritmos genéticos mediante la aplicación de alguna función de mapeo (decodificación) de la representación del cromosoma en el espacio de las variables de decisión. De esta manera, cada una de las cadenas contenidas en el cromosoma se decodifican para formar un vector de orden N_{par} de acuerdo al número de dimensiones en el espacio de búsqueda.

Las variables de decisión se almacenan en una matriz numérica de tamaño NxN_{par}. De esta manera, cada uno de los renglones de la matriz corresponde a un fenotipo de un cromosoma particular. En el caso de tener una sola variable de decisión, el tamaño de la matriz va a ser de $Nx1$, que finalmente es un vector. Este fenotipo está representado en el algoritmo por la variable *parámetro.*

Valor de la Función Objetivo. La función objetivo es usada para dar una medida de como se han desempeñado los individuos en el dominio del problema. En el caso de un problema de minimización, el individuo más apto tendrá el valor numérico más bajo de la función de adaptabilidad (*fitness*) asociada. Esta medida de adaptabilidad es usada solo como una etapa intermedia en la determinación del desempeño relativo de los individuos en un Algoritmo Genético. Por otro lado, la función de adaptabilidad es normalmente usada para transformar el

valor de la función objetivo en una medida de adaptación relativa [2][3][6][8] o normalización de este valor, por lo tanto:

$$F(x) = g(f(x)) \tag{3.1}$$

donde f es la función objetivo, g transforma el valor de la función objetivo a un número no negativo y F es la adaptabilidad relativa resultante. Este mapeo es siempre necesario cuando la función objetivo debe minimizarse, con lo cual el valor más bajo de la función objetivo corresponde al individuo más apto. En muchos casos, el valor de adaptabilidad corresponde al número de descendientes que un individuo puede esperar para producir la siguiente generación.

Los valores de la función objetivo pueden ser escalares o, en el caso de un problema multiobjetivo, vectoriales. Cabe hacer notar que los valores de la función objetivo no necesariamente son los mismos valores de la función de adaptabilidad.

Los valores de la función objetivo son almacenados en una matriz numérica de tamaño NxN_{obj}, donde N_{obj} es el número de objetivo. Esta estructura está representada por la variable x.

Valor de la Función de Adaptabilidad. Los valores de adaptabilidad son derivados de los valores de la función objetivo. Estos son valores escalares no negativos y son almacenados en un vector columna de longitud N, representado por la variable *fitness*.

Además de estos datos, se puede incluir información adicional. En la figura 3.1. se presenta un esquema de esta estructura.

Numero de Individuo	Individuos			
	Cadena	x	f(x)	Otros
1	01111	15	225	
2	01001	9	81	
3				
n	00111	7	49	

Figura 3.1. Estructura de Datos para el AG.

En el Algoritmo Genético Simple (AGS) se aplican los operadores genéticos básicos de cruzamiento y mutación a la población presente para producir una nueva población y pasar de generación en generación. Como su contraparte natural, el operador de cruzamiento debe producir

nuevos individuos que tienen partes del material genético de ambos padres.

Este operador produce el intercambio de información entre parejas de individuos escogidos por alguno de los métodos de selección de la población presente. La forma más simple de cruzamiento es el cruzamiento en un solo punto [1][3][8], aunque existen otras maneras de realizar el cruzamiento como se describió en el capítulo II.

Para esta implementación se genera un número aleatorio entre [0,1]. Si este número es menor o igual a la probabilidad de cruzamiento P_c definida, se genera el intercambio de información tomando como punto de cruzamiento el punto determinado aleatoriamente entre **[1,L-1]**, donde L es la longitud de la cadena o cromosoma artificial.

Para el operador de mutación, nosotros tenemos que en la evolución natural, la mutación es un proceso aleatorio donde un alelo de un gen es reemplazado por otro para producir una nueva estructura genética. De la misma manera que para el operador de cruzamiento, si el número aleatorio entre [0,1] es menor o igual a la probabilidad de mutación P_m, una posición del cromosoma (la posición en el momento del cruzamiento) cambiará su estado: si es 1 a 0 y si es 0 a 1.

Para producir una nueva población y pasar de generación en generación se definen dos poblaciones, representadas como *pop (crom, fitness) y popn (cromn, fitnessn)*, las cuales corresponden a la población presente y a la nueva población, respectivamente (Figura 3.2.).

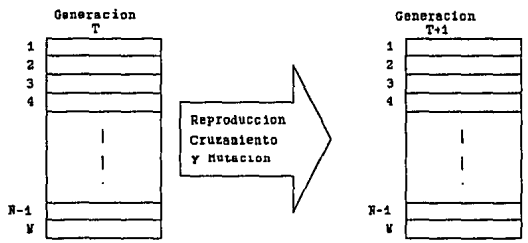


Figura 3.2. Evolución de Poblaciones.

Con la estructura de datos definida, los esquemas de reproducción y los operadores genéticos básicos, se definen las variables globales del programa que afectan el desarrollo del algoritmo en cada generación. Estas variables son:

* *pcros*: probabilidad de cruzamiento.
* *pmutacion*: probalidad de mutación.
* *popsize*: tamaño de la población.
* *lcrom*: longitud del cromosoma.
* *maxgen*: número máximo de generaciones de ejecución del algoritmo.
* *sumfitness*: suma de los valores de la función de adaptabilidad de la población.
* *avg*: valor de la función de adaptabilidad promedio.
* *max*: valor máximo de la función de adaptabilidad.
* *min*: valor mínimo de la función de adaptabilidad.

De acuerdo a las características de los Algoritmos Genéticos, los operadores genéticos pueden implementarse de una manera directa. Cabe señalar que estos algoritmos son métodos de búsqueda aleatoria, por lo cual el apéndice A presenta tres rutinas que generan número aleatorios definiéndose de la siguiente manera:

* *rand*: Función propia del intérprete **MATLAB (Matrix Laboratory)**, la cual regresa un número aleatorio entre '0' y '1'.
* *flipm*: Esta función regresa un valor de '0' o '1' dependiendo de una probabilidad especificada.
* *rnd*: Función que regresa un valor entero entre los límites superior e inferior especificados.

III.2.1. Criterio de Terminación de un Algoritmo Genético.

Como los Algoritmos Genéticos son métodos de búsqueda estocástica, es difícil especificar formalmente un criterio de convergencia. Como la adaptabilidad de una población puede quedar estática para un número de generaciones antes de que un individuo superior sea encontrado, la aplicación de un criterio de terminación convencional es problemático. Una práctica común es terminar el algoritmo después de un número predefinido de generaciones y analizar la calidad de los mejores miembros de la población respecto a la definición del problema. Si las soluciones encontradas no son aceptables, el algoritmo puede ser reiniciado.

III.3. FORMULACION DEL PROBLEMA.

Para ilustrar el desempeño de los diferentes esquemas de selección y el comportamiento del algoritmo a las variaciones de las probabilidades de los operadores genéticos, se implementó un algoritmo genético analizando una función exponencial simple definida de la siguiente manera:

$$f(x) = e^{(n*\ln(x/coef))} \tag{3.2}$$

donde:

coef	es igual a 2^L.
n	es igual a 10.
x	es el valor de la cadena decodificada.
L	es la longitud del cromosoma o cadena.

En la figura 3.3. se representa esta función observando una exponencial creciente en la cual la solución óptima corresponde al valor máximo de *x*. Esto es, $x=2^L$. Esta función exponencial creciente nos representa la función objetivo. En este caso en particular la función f(x) representa además de la función objetivo, la función de adaptabilidad.

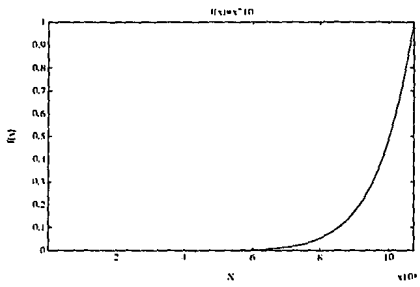


Figura 3.3. Función Objetivo.

Esta función es implementada tomando un cromosoma de longitud $L=30$, con lo cual nuestro espacio de soluciones o dominio del problema está formado por 2^L posibles valores. Este va a ser el único parámetro que se va a conservar constante, ya que tanto las probabilidades de los operadores genéticos así como el tamaño de la población van a tomar diferentes valores.

III.4. METODOS DE SELECCION.

En los algoritmos genéticos el proceso de selección o esquema de reproducción tiene como función la selección de individuos de una población presente de acuerdo a su nivel de

desempeño o adaptabilidad en el dominio del problema para producir conjuntamente con los operadores genéticos, una nueva población con mejores individuos que los individuos originales. La selección de individuos puede verse como dos procesos separados:

1) Determinación del número de veces que un individuo espera ser seleccionado.
2) Conversión del número esperado de selecciones en un número discreto de descendientes.

La primera está relacionada con la transformación de los valores de adaptabilidad en un número de esperanza real de la probabilidad de un individuo a reproducir. La segunda es la selección probabilística de los individuos para la reproducción basados en la adaptabilidad de los individuos. Esto también se conoce como *muestreo* [6].

De estos esquemas de reproducción analizaremos tres métodos: *el método de la ruleta [3][4][8], el método de escalamiento lineal [3][4] y el método ranking lineal y no lineal [2][3][6].*

III.4.1. Método de la Ruleta.

Este esquema de reproducción asigna a cada uno de los miembros de la población una sección de la *ruleta* en función de su desempeño. Posteriormente, la ruleta se gira y se detiene aleatoriamente determinando así el individuo de la población seleccionado. Este proceso se realiza *N* veces (donde *N* es el número de elementos de la población) para determinar que elementos van a contribuir en la formación de la nueva generación.

Basados en el código presentado en el apéndice A, este método de selección, representado en la función *select*, regresa el índice que corresponde al miembro de la población seleccionado. Esto se realiza tomando la suma parcial de los valores de la función a evaluar acumulados en la variable real *partsum*. La variable *rando* contiene la posición donde la ruleta debe detenerse después de *N* giros aleatorios de acuerdo a la expresión

rando := rand * sumfitness

La variable *sumfitness* calculada en el procedimiento *estadístico*, es multiplicada por un número generado aleatoriamente por la función *rand*. Finalmente, el constructor *while* busca en los valores de la ruleta hasta que la suma parcial es mayor o igual a la variable *rando*, regresando el índice del miembro de la población que nos lleva a este resultado.

La figura 3.4. muestra el comportamiento del AG con la función objetivo exponencial utilizando este método de selección. El análisis es realizado para diferentes probabilidades de cruzamiento y mutación, manteniéndose constante el tamaño de la población en *popsize=30*. En

la figura 3.4.a. $P_c=0.5$ variando la probabilidad de mutación. Para las figuras 3.4.b. y 3.4.c la probabilidad de cruzamiento es de *0.75* y *1.0*, respectivamente.

En estas gráficas podemos observar que para probabilidades bajas de mutación se tiende a estabilizar el valor de *fitness*, pero este valor difiere del valor óptimo (convergencia prematura). Por el contrario, los valores altos de P_m ocasionan que el **AG** llegue al valor óptimo de *fitness* pero nunca se llega a estabilizar. Las probabilidades altas de mutación generan tal diversidad en la población que el **AG** empieza a parecerse a una denominada *búsqueda aleatoria.*

En el caso del operador de cruzamiento observamos mejores resultados para $P_c=0.75$ y $P_c=1.0$. La probabilidad de $P_c=0.5$ nos indica que la mitad de las parejas seleccionadas no van a presentar un intercambio de información, manteniéndose sin evolucionar, modificándose ocasionalmente por el operador de mutación.

Al usar este método de selección, es poco probable que el número de individuos de un denominado esquema en la población refleje el calculado teóricamente. Existe una fuente inherente de error de muestreo, el cual combinado con la variancia asociada con los procesos estocásticos, nos lleva a desviaciones considerables en el número de individuos teóricamente predecidos por el *Teorema del Esquema* [7]. El efecto de estos errores puede ser tal que algunos individuos con esquemas importantes puedan desaparecer de la población.

En la figura 3.5. se muestra el comportamiento del **AG** con este mismo esquema de selección para diferentes tamaños de la población, con $P_c=0.75$ y $P_m=0.01$. Se observa que aunque en pocas generaciones cada una de las poblaciones mantiene su mejor valor de fitness arriba del 90 porciento, las poblaciones más grandes tienden a estar más cercanas al valor óptimo. Sin embargo, todas presentan una mala convergencia.

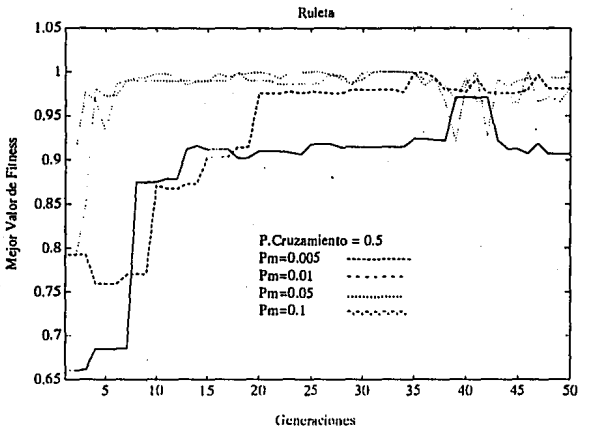


Figura 3.4.a. Método de la Ruleta (P_c=0.5).

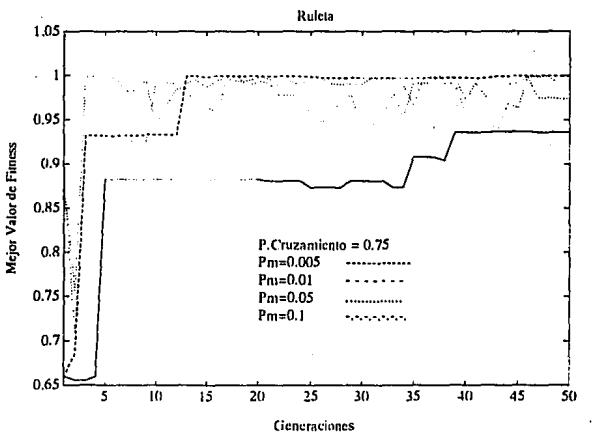


Figura 3.4.b. Método de la Ruleta (P_c=0.75).

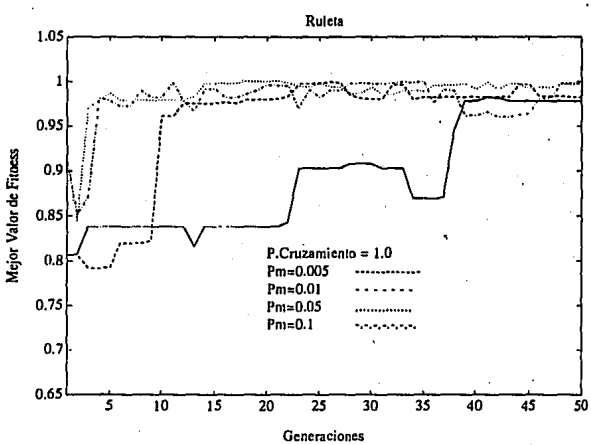


Figura 3.4.c. Método de la Ruleta (P_c=1.0).

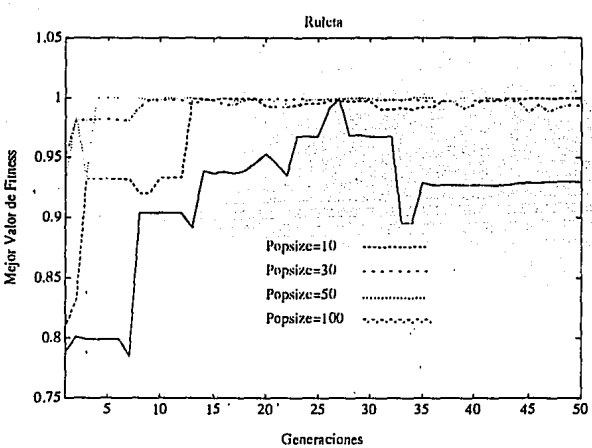


Figura 3.5. Método de la Ruleta (popsize variable).

III.4.2. Método de Escalamiento Lineal.

Cuando se tienen algoritmos genéticos con poblaciones pequeñas, es importante regular el número de copias con las que cada uno de los miembros de la población presente van a contribuir a la siguiente generación. Al iniciar el algoritmo, con una población pequeña, se presenta el caso del predominio de los mejores elementos de esa generación (sin ser necesariamente el óptimo), llevándonos a una convergencia prematura, es decir todos los elementos de la población tienden a ser iguales. Lo anterior provoca que elementos con valores de la función objetivo bajos pero con información importante, no puedan contribuir en generaciones sucesivas mejorando el valor de la función. Por lo anterior, es importante conservar los niveles apropiados de competencia entre los miembros de la población; ésto es, conservar la diversidad de la población.

El *método de escalamiento* altera el valor de la función conservando el valor promedio y haciendo que los miembros de la población con valores arriba del promedio tiendan a bajarlo. Esto provoca que los miembros promedio y los mejores elementos presenten el mismo número de copias en generaciones futuras, teniendo intercambio de información más diversa.

El método de escalamiento requiere una relación lineal entre f' y f, siendo f el valor de la función objetivo y f' el valor escalado:

$$f' = af + b \tag{3.3}$$

Los coeficientes **a** y **b** pueden elegirse de diferentes maneras, sin embargo, en todos los casos se requiere que f'_{avg} (promedio del valor escalado) sea igual a f_{avg} (promedio del valor real). Para controlar el número de descendientes otorgado al miembro de la población más apto, se presenta otra relación de escalamiento para obtener un valor de la función del mejor miembro escalado, $f'_{max}=C_{mult}*f_{avg}$, donde C_{mult} es el número de copias esperadas para el mejor elemento de la población. El valor de C_{mult} está típicamente en el rango de [1.2, 2.0] [3].

Este método puede resumirse en la gráfica presentada en la figura 3.6. Como se puede observar, los mejores elementos tienden a escalarse reduciendo su valor, y los elementos bajos se escalan incrementando su valor. Todos los elementos tienden a acercarse al valor promedio al aplicar el escalamiento. En la figura 3.7. se presenta un caso en el cual el método de escalamiento no es aplicable porque provoca que los valores más bajos se hagan negativos. Esto se presenta cuando los elementos menos aptos están muy por debajo del promedio de la población y el valor máximo de la misma.

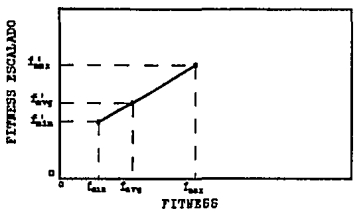


Figura 3.6. Escalamiento Lineal.

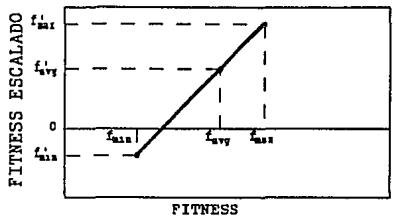


Figura 3.7. Escalamiento Lineal.

El procedimiento de escalamiento puede implementarse fácilmente y adicionarlo al código del algoritmo genético simple mediante tres rutinas descritas en el apéndice A:

* **prescala**
* **escala**
* **escalapop**

La rutina de preescalamiento calcula los valores de los coeficientes **a** y **b** a partir de los valores máximo, mínimo y promedio de la función objetivo. Esto es:

Si min > (C_{mult}*avg - max)/(C_{mult} - 1.0)
delta = max - avg
a = (C_{mult} - 1.0) * avg/delta
b = avg * (max - C_{mult}*avg)/delta
Si no
delta = avg - min
a = avg/delta
b= -min*avg/delta

donde el valor de C_{mult} está en el rango de [1.2,2.0]

El procedimiento de escala población, **'escalpop'**, es llamado después del preescalamiento. En este procedimiento se calcula el nuevo valor de la función a partir del uso de la ecuación (3.2) definida en la función **escala**, así como el nuevo valor de la variable **sumfitness.**

Con el nuevo valor de los elementos de la población y el valor de la variable **sumfitness**, se procede a aplicar el método de la ruleta a estos valores. De esta manera, el escalamiento lineal ayuda a prevenir la convergencia prematura de la población.

La figura 3.8. muestra la implementación de un AG utilizando el método de la Ruleta y Escalamiento Lineal de valores de *fitness*. Realizando las misma variaciones en las probabilidades

de los operadores genéticos de cruzamiento y mutación, se presentan las figuras 3.8.a., 3.8.b. y 3.8.c. que corresponden a probabilidades de cruzamiento de 0.5, 0.75 y 1,0, respectivamente, variando en los tres casos la probabilidad de mutación.

En estas tres gráficas podemos observar una mejor respuesta en el comportamiento del **AG**, en comparación con el esquema de reproducción de la Ruleta.

Con probabilidades bajas de mutación, el algoritmo toma más generaciones en alcanzar un valor muy cercano al óptimo. También para una probabilidad de cruzamiento de $P_c=0.5$, toma más generaciones en alcanzar el valor óptimo o un valor muy cercano a éste. Pero el escalamiento lineal aplicado a la Ruleta, evita el dominio de algunos de los miembros de la población con valores altos de *fitness* manteniendo niveles de competencia adecuados.

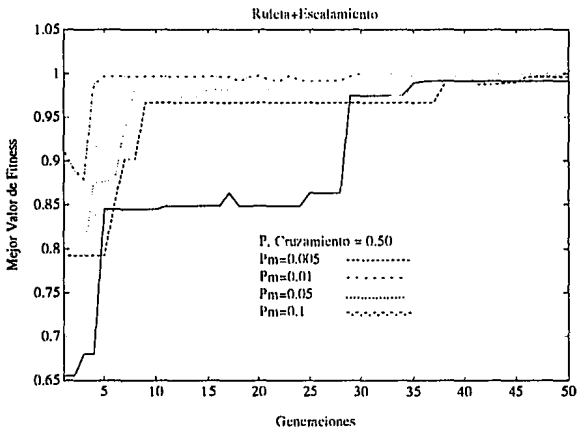


Figura 3.8.a. Método de Escalamiento (P_c=0.5).

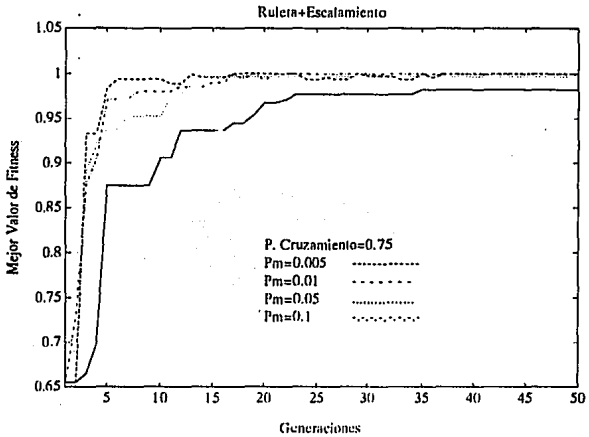


Figura 3.8.b. Método de Escalamiento (P_c=0.75).

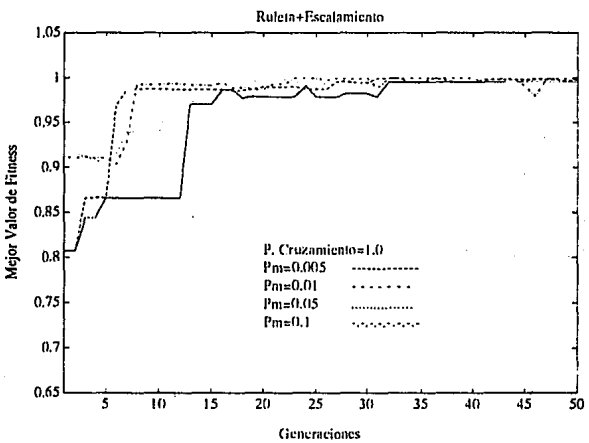
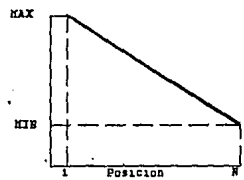


Figura 3.8.c. Método de Escalamiento (P_c=1.0).

III.4.3. Método Ranking.

Este método calcula el número de descendientes de cada uno de los elementos de la población de acuerdo a la posición que presenta cuando se han ordenado los elementos en forma descendiente (Figura 3.9.). Esta asignación del número de copias puede realizarse de manera lineal o no lineal.



Esquema de Asignacion en el esquema Ranking.

Figura 3.9. Método Ranking.

Cuando se tienen el caso del método **ranking lineal** [2][6], se presenta de alguna manera un escalamiento. La varaible **MAX**, es usada para determinar la **presión selectiva** [6]; ésto nos lleva a determinar la función de adaptabilidad aplicando las siguientes reglas:

- **MIN = 2.0 - MAX**
- **INC = 2.0 * (MAX - 1.0)/N**
- **LOW = INC/2.0**

donde **MIN** es el límite inferior, **INC** es la diferencia entre el valor de adaptabilidad de individuos adyacentes y **LOW** es el número esperado de veces que el individuo menos apto de la población es seleccionado. **MAX** es elegido en el intervalo de [1.1, 2.0]. De allí que, para una población de tamaño **N=40** y **MAX**=1.1, se obtiene **MIN**=0.9, **INC**=0.05 y **LOW**=0.025. El nuevo valor de adaptabilidad de los individuos en una población pueden ser calculados directamente de:

$$f(x_i) = MIN + (MAX - MIN)\frac{x_i - 1}{N - 1} \tag{3.3}$$

substituyendo el valor de **MIN**, tenemos:

$$f(x_i) = 2 - MAX + 2(MAX - 1)\frac{x_i - 1}{N - 1} \tag{3.4}$$

donde x_i es la posición en la población ordenada del individuo i.

Así como el método de escalamiento, la principal ventaja de este método de selección es el hecho de que el algoritmo es menos propenso a la convergencia prematura causada por los individuos que están muy por debajo del promedio cuando se trata de un problema de maximización, y por los individuos muy por encima del valor promedio en el caso de minimización.

Para el caso del método **ranking no lineal** [2], éste puede ser implementado de muchas maneras. La rutina para el método ranking no lineal del apéndice A, crea un vector en el cual se tienen los índices de individuos tomados de una lista ordenada de miembros de una población. La frecuencia de un individuo corresponde al número de copias o descendientes esperados del mismo. Esto se realiza de la siguiente manera:

1. Ordenar la población descendentemente de acuerdo al valor de fitness.
2. Asignar una copia al valor de adaptabilidad promedio (índice del individuo promedio).
3. En base al valor de la variable MAX, el cual determina el número de copias para el individuo más apto, se determina el número de copias de cada uno de los individuos que están arriba del promedio de manera lineal.
4. Si el número de copias asignadas hasta este punto es menor a N, se asigna una copia a cada uno de los individuos abajo del promedio asignando el índice de estos individuos al vector hasta que el número de copias asignadas en total es igual a N.
5. En este momento se tiene un vector con los índices, de acuerdo a la población ordenada, de los individuos que se les asignaron copias. Por ejemplo, el elemento 0 del vector va a tener el índice del individuo que presenta el valor promedio, y si MAX es igual a 3, los elementos del 1 al 3 del vector van a tener el índice del mejor elemento que en este caso, con la población ordenada, es el índice 0.

Otra manera de implementar el método ranking no lineal es asignar arbitrariamente el número de copias a cada uno de los miembros de la población, tomando en consideración que el mejor elemento va a tener el máximo número de copias.

Lo anterior puede realizarse al definir previamente un vector de tamaño *N*, el cual contiene los índices que corresponden a los miembros de la población considerándola ordenada. Estos índices pueden repetirse, lo cual nos va a indicar el número esperado de copias para la siguiente generación de ese individuo. Este vector de índices guarda las mismas características que el vector del paso 5.

La figura 3.10. muestra el comportamiento del AG para el método de selección Ranking. Al igual que en los métodos de la Ruleta y Escalamiento Lineal, se variaron las probabilidades de cruzamiento y mutación y dejando constante el tamaño de la población con *popsize=30.*

El efecto de variar P_m y P_c se ilustra en las figuras 3.10.a., 3.10.b. y 3.10.c. Se observa que para todas las probabilidades de mutación y cruzamiento, la estabilidad del valor de ***fitness*** es muy alta, es decir, se mantiene por lo menos en el valor de la generación anterior. Con este esquema de reproducción, aunque la convergencia del algoritmo todavía depende de la población inicial, el número de generaciones en las cuales alcanza el valor óptimo se ha decrementado considerablemente, y para un valor de P_m=*0.05* y P_m=*0.1*, la convergencia al óptimo se obtiene en menos de cinco generaciones.

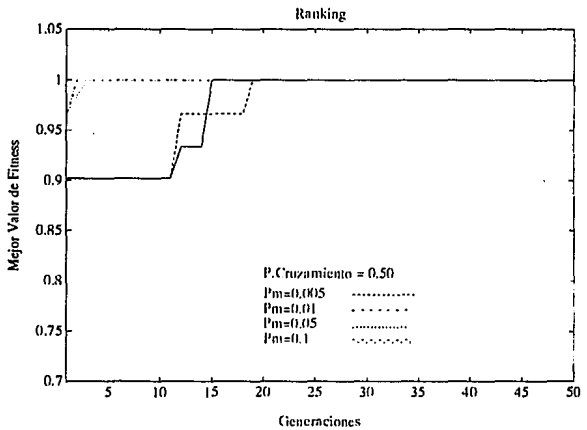


Figura 3.10.a. Método Ranking (P_c=0.5).

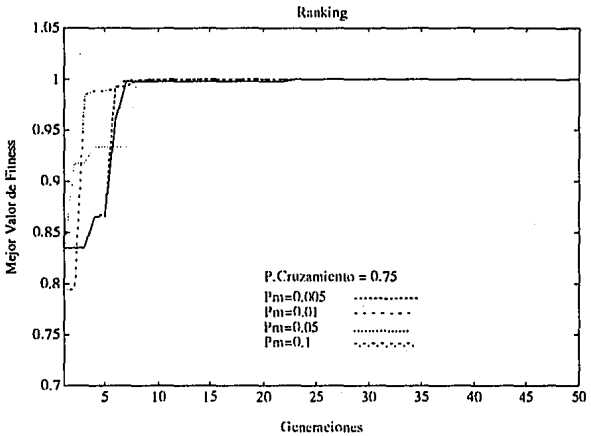


Figura 3.10.b. Método Ranking (P_c=0.75).

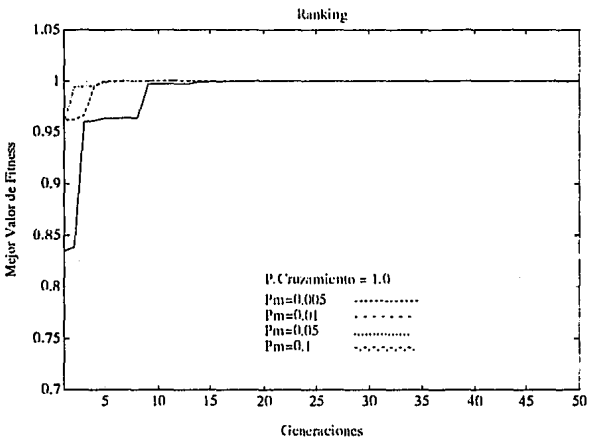


Figura 3.10.c. Método Ranking (P_c=1.0).

En la figura 3.11. se ilustra el efecto de utilizar este esquema de reproducción para diferentes tamaños de la población, manteniendo constantes $P_c=1.0$ y $P_m=0.1$. Las gráficas muestran que aún cuando *Ranking* controla la diversidad en la población, si ésta es pequeña, el algoritmo puede tomar muchas generaciones para converger dependiendo de la diversidad inicial de la población. Lo opuesto ocurre para poblaciones grandes en donde la tendencia es a converger en tan solo pocas generaciones. Otro aspecto importante a notar es que el valor de *fitness* se incrementa monóticamente hasta llegar al óptimo.

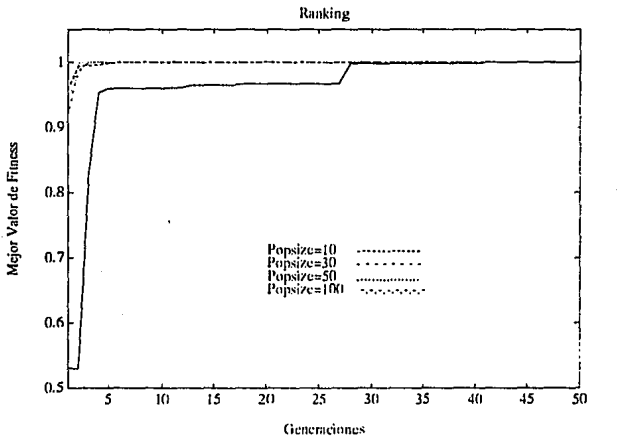


Figura 3.11. Método Ranking (popsize variable).

III.5. CONCLUSIONES.

Los *Algoritmos Genéticos*, basados en los mecanismos de la evolución natural son métodos simples, robustos y eficientes, los cuales poseen diversos parámetros que son fundamentales en su desempeño: *la probabilidad de mutación, la probabilidad de cruzamiento, el tamaño de la población y el método de selección elegido.*

La correcta selección del esquema de reproducción empleado para determinar los elementos de la población a reproducirse así como el uso de adecuadas probabilidades de cruzamiento y mutación, es crítico en la calidad de convergencia del algoritmo y por lo tanto es relevante en el tiempo empleado para llegar a una solución óptima.

La probabilidad de mutación se encuentra ligada, en cierta manera, a la longitud del cromosoma que se esté empleando. En este análisis, la longitud del cromosoma es de $L=30$, con lo cual se podría establecer una probabilidad de mutación de $P_m=1/30=0.0333$. De acuerdo a los resultados, para los métodos de la Ruleta y Escalamiento Lineal, se presentó un mejor desempeño del algoritmo con probabilidades de mutación establecidas en el rango de [0.01,0.05].

El tamaño de la población elegida para el algoritmo no puede ser muy pequeña, ya que no presenta una diversidad en los elementos de la población y ésto toma un número mayor de generaciones en alcanzar el valor óptimo o muy cercano a éste. Por el contrario, una población grande nos presenta una mayor diversidad y en pocas generaciones podemos llegar al valor óptimo, pero nos toma más tiempo en pasar de una generación a otra. Por lo tanto debemos tomar una población de tamaño tal que se tenga una diversidad adecuada de la población sin tener un alto número de individuos.

III.6. REFERENCIAS.

[1] Eshelman, L.J., Caruana, R.A., Schaffer, J.D., "Biases in the Crossover Landscape", Proceedings of the Third International Conference on Genetic Algorithms, San Mateo Cal: Morgan Kaufmann Publishers, pp. 10-19, 1989.

[2] Fleming, P.J., Chipperfield, A., Pohlheim, H., Fonseca, C., "Genetic Algorithms Toolbox for Use with Matlab., Department of Automatic Control and Systems Engineering, University of Sheffield, U.K., 1994.

[3] Goldberg, D.E., "Genetic Algorithms in Search, Optimization and Machine Learning, MA: Addison-Wesley, 1989.

[4] Goldberg, D.E., Kalyanmoy, D., "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms", Foundations of Genetic Algorithms, IEEE Computer Society Press, Los Alamitos, Cal., pp. 69-93, 1992.

[5] Goldberg, D.E., "Sizing Populations for Serial and Parallel Genetic Algorithms", Proceedings of the Third International Conference on Genetic Algorithms, San Mateo, Cal: Morgan Kayfmann Publishers, pp. 70-79, 1989.

[6] Grefenstette, J.J., Baker, J.E., " How Genetic Algorithms Work: A Critical Look at Implicit Parallelism", Proceedings of the Third International Conference on Genetic Algorithms, San Mateo, Cal: Morgan Kaufmann Publishers, pp. 20-27, 1989.

[7] Holland, J.J., "Adaptation in Natural and Artificial Systems", Ann Arbor, MI: University of Michigan Press, 1975.

[8] Lawrence, D., "Handbook of Genetic Algorithms", Van Nostrand Reinhold, N. Y., 1991.

[9] User's Guide Matlab.

CAPITULO IV

ESTIMACION ESPECTRAL PARAMETRICA UTILIZANDO ALGORITMOS GENETICOS

El objetivo de este capítulo es presentar un camino alternativo para la implementación de un estimador espectral paramétrico, con la finalidad de reducir la complejidad que presentan dichos estimadores en relación con los métodos basados en la Transformada de Fourier.

Tomando las bases del Algoritmo Genético desarrollado en el capítulo anterior, podemos directamente implementar el estimador paramétrico al minimizar la función objetivo, la cual corresponde a la función de error definida por el estimador elegido.

IV.1. INTRODUCCION.

Los métodos clásicos de estimación espectral (métodos basados en la Transformada Rápida de Fourier) hacen uso de ventanas de datos o valores estimados de la *Función de Autocorrelación ACF* [5]. Esto nos hace suponer implícitamente que, los datos no observados o los valores de la ACF fuera de la ventana son cero, lo cual generalmente, no es una suposición real. Frecuentemente se tiene un mayor conocimiento acerca del proceso del cual fueron tomados los datos, o al menos se hace una suposición más razonable que la de tomar los datos o valores de la ACF fuera de la ventana como cero. El uso de información previa puede permitir la selección de un modelo más exacto para el proceso que generó la muestra de datos, o un modelo el cual es una mejor aproximación al proceso base. Es posible entonces obtener una mejor estimación espectral basándose en dicho modelo y estimar sus parámetros a partir de las observaciones. Así, la estimación espectral, en el contexto del modelado, se puede resumir en un proceso de tres pasos. El primero es seleccionar el modelo, el segundo es estimar los parámetros del modelo elegido usando los datos disponibles, y el tercer paso consiste en obtener el espectro substituyendo los parámetros estimados del modelo en la función de *Densidad de Potencia Espectral (PSD)* teórica definida por el modelo.

Otro aspecto por el cual se tiene interés por el modelado paramétrico es el hecho de que los métodos de estimación espectral modernos presentan una mejor resolución que los métodos convencionales. El grado de mejoramiento en la resolución y fidelidad del espectro estará determinado por la capacidad de adaptar un modelo con pocos parámetros a los datos observados.

Una de las principales características del modelado paramétrico en la estimación espectral, es la disposición que se tiene de poder hacer suposiciones más reales sobre la naturaleza del proceso medido fuera del intervalo de medición, en lugar de suponer que son ceros o cíclica, eliminando el uso de funciones de ventanas.

IV.2. MODELADO PARAMETRICO.

Como se mencionó anteriormente, el modelado paramétrico para la estimación espectral consiste en elegir un modelo apropiado, estimar los parámetros del modelo, y entonces, substituir estos valores estimados en la expresión de la Densidad Espectral de Potencia Teórica [5][6][7]. Dentro del modelado paramétrico se presentan tres tipos de modelos: *Autorregresivo (AR), movimiento promedio (MA), y autorregresivo-movimiento promedio (ARMA).*

El diagrama de bloques general de este proceso de modelado paramétrico se presenta en la figura 4.1.

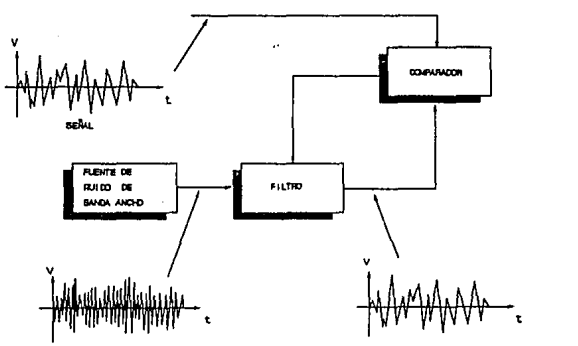


Figura 4.1. Proceso de Modelado Paramétrico.

En el diagrama podemos observar que se tiene un sistema en lazo cerrado el cual toma ciertos datos de entrada (señal aleatoria). Dicha señal es pasada a través de un filtro para obtener una señal estimada la cual es comparada contra la señal real. Los resultados obtenidos de esta comparación representan una función de error a minimizar. Estos resultados son tomados por el

filtro para determinar los reajustes de los parámetros del mismo que generen una señal estimada más cercana a la real y por lo tanto, reduzca el error entre ambas señales. Los parámetros del filtro están relacionados con el orden y tipo de modelo del estimador, el cual puede ser AR, MA o ARMA.

IV.3. ESTIMACION ESPECTRAL UTILIZANDO EL MODELO AR.

IV.3.1. Introducción.

Dentro de los métodos de modelado paramétrico el más popular de ellos es el estimador espectral autorregresivo AR, el cual es utilizado en este trabajo. Esto se debe a que la estimación de los parámetros pueden encontrarse al resolver un conjunto de ecuaciones lineales. Para la estimación de los parámetros de los modelos ARMA y MA, es necesario resolver un conjunto de ecuaciones no lineales. Otros nombres con los cuales también es conocido el estimador espectral AR son *estimador espectral de máxima entropía* y *estimador espectral de predicción lineal* [5].

En los procesos AR, el problema de predicción lineal está relacionado con la predicción de datos no observados x[n]. Para tal efecto se presenta la predicción lineal en adelanto y la predicción lineal en atraso.

La predicción en adelanto está definida como:

$$\hat{x}^f[n] = -\sum_{i=1}^{m} a^f[i]x[n-i] \tag{4.1}$$

donde $a^f[i]$ son los coeficientes de predicción en adelanto. La predicción $x^f[n]$ está basada en la secuencia de datos observados ($x[n-1]$, ..., $x[n-k]$) de las últimas k muestras.

Una medida de la calidad de esta estimación está dada por el error predictivo en adelanto:

$$e^f[n] = x[n] - \hat{x}^f[n] \tag{4.2}$$

la cual es frecuentemente usada en la forma de error cuadrático medio. Esto es:

$$\mathcal{E}[\,|e^f[n]|^2] = \mathcal{E}[\,|x[n]-\hat{x}^f[n]|^2] = p^f \tag{4.3}$$

La cantidad p^f es conocida como el error de predicción en adelanto y es una medida de la variancia. Para calcular los coeficientes $a^f[i]$ se requiere calcular el valor mínimo de p^f [6][9], con

lo cual, al desarrollar (4.3) se puede obtener el vector $a^f[i]$ el cual minimiza la variancia p^f obtenida al resolver el sistema matricial

$$\begin{bmatrix} r_{xx}[0] & r_{xx}[-1] & \cdots & r_{xx}[-k] \\ r_{xx}[1] & r_{xx}[0] & \cdots & r_{xx}[-(k-1)] \\ & & \cdots & \\ r_{xx}[k] & r_{xx}[k-1] & \cdots & r_{xx}[0] \end{bmatrix} \begin{bmatrix} 1 \\ a^f[1] \\ \cdots \\ a^f[k] \end{bmatrix} = \begin{bmatrix} p^f \\ 0 \\ \cdots \\ 0 \end{bmatrix} \tag{4.4}$$

Ahora bien, la predicción lineal en atraso está definida de manera similar a la predicción en adelanto solo que ahora tomando k muestras después. Esto es:

$$\hat{x}^b[n] = -\sum_{i=1}^{k} a^b[i]\,x[n+i] \tag{4.5}$$

El error predictivo en atraso esta definido por:

$$e^b[n] = x[n-k] - \hat{x}^b[n-k] \tag{4.6}$$

$$p^b = \mathcal{E}\left[\,|e^b[n]|^2\right] \tag{4.7}$$

Análogamente, el cálculo de la secuencia $a^b[i]$ que minimiza p^b, es obtenida al resolver el sistema:

$$\begin{bmatrix} r_{xx}[0] & r_{xx}[-1] & \cdots & r_{xx}[-k] \\ r_{xx}[1] & r_{xx}[0] & \cdots & r_{xx}[-(k-1)] \\ & & \cdots & \\ r_{xx}[k] & r_{xx}[k-1] & \cdots & r_{xx}[0] \end{bmatrix} \begin{bmatrix} a^b[k] \\ a^b[k-1] \\ \cdots \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \cdots \\ p^b \end{bmatrix} \tag{4.8}$$

Cuando se resuelven (4.4) y (4.8), se definen las siguientes propiedades [6]:

$$p^b = p^f$$
$$a^b[i] = (a^f[i])^*$$

para $1 \le i \le k$, lo cual significa que para filtros de predicción con la misma longitud k, los coeficientes de predicción lineal en atraso son iguales al conjugado complejo de los coeficientes

de predicción lineal en adelanto.

IV.3.2. Métodos de Estimación Espectral AR.

Para la estimación de la densidad espectral de potencia autorregresiva (PSD AR), se presentan una variedad de métodos entre los cuales se tienen el de Autocorrelación o de Yule-Walker, de *Covariancia Modificada* y de Burg [5][6]. Generalmente, la calidad del espectro estimado es muy buena, aun para secuencias cortas de datos. Para muchos de los métodos, los requerimientos computacionales pueden reducirse si se toman las ventajas de la estructura de las ecuaciones a ser resueltas para la implementación de algoritmos "rápidos".

Estudios previos [9] han investigado el desempeño de diversos estimadores espectrales paramétricos AR, identificando el método de ***Covariancia Modificada*** como el más conveniente en términos de eficiencia y complejidad computacional.

Algoritmo de Covariancia Modificada.

El método paramétrico de estimación espectral autorregresivo de *covariancia modificada* calcula los parámetros del modelo al minimizar el error predictivo en adelanto y atraso promedio *(ρ)*, haciendo uso de todos los datos disponibles (x[0], ..., x[N]),

$$\hat{\rho} = \frac{1}{2} \cdot (\hat{\rho}^f + \hat{\rho}^b) \tag{4.9}$$

donde el error predictivo en adelanto (ρ^f) y el error predictivo en atraso (ρ^b) están dados por:

$$\hat{\rho}^f = \frac{1}{N-p} \sum_{n=p}^{N-1} \left| x[n] + \sum_{k=1}^{p} a[k]x[n-k] \right|^2 \tag{4.10}$$

$$\hat{\rho}^b = \frac{1}{N-p} \sum_{n=0}^{N-1-p} \left| x[n] + \sum_{k=1}^{p} a^*[k]x[n+k] \right|^2 \tag{4.11}$$

Para minimizar la ecuación 4.16, podemos diferenciar ρ con respecto a las partes real e imaginaria de $a[k]$ para $k=1,2,...,p$,

$$\frac{\partial \hat{\rho}}{\partial a[l]} = \frac{1}{N-p}\left[\sum_{n=p}^{N-1}\left(x[n]+\sum_{k=1}^{p} a[k]x[n-k]\right)x^*[n-l]\right.$$

$$\left.+\sum_{n=0}^{N-1-p}\left(x^*[n]+\sum_{k=1}^{p} a[k]x^*[n+k]\right)x[n+l]\right]=0, \qquad l=1,2,..p. \tag{4.12}$$

Simplificando:

$$\sum_{k=1}^{p} \hat{a}[k]\left(\sum_{n=p}^{N-1} x[n-k]x^*[n-l]+\sum_{n=0}^{N-1-p} x^*[n+k]x[n+l]\right)$$

$$= -\left(\sum_{n=p}^{N-1} x[n]x^*[n-l]+\sum_{n=0}^{N-1-p} x^*[n]x[n+l]\right) \tag{4.13}$$

Entonces, el error predictivo óptimo ρ_{min} es obtenido al resolver el sistema de ecuaciones de covariancia modificada dado por:

$$\begin{bmatrix} c_{xx}[1,1] & c_{xx}[1,2] & \cdots & c_{xx}[1,p] \\ c_{xx}[2,1] & c_{xx}[2,2] & \cdots & c_{xx}[2,p] \\ \vdots & \vdots & \ddots & \vdots \\ c_{xx}[p,1] & c_{xx}[p,2] & \cdots & c_{xx}[p,p] \end{bmatrix} \cdot \begin{bmatrix} \hat{a}[1] \\ \hat{a}[2] \\ \vdots \\ \hat{a}[p] \end{bmatrix} = - \begin{bmatrix} c_{xx}[1,0] \\ c_{xx}[2,0] \\ \vdots \\ c_{xx}[p,0] \end{bmatrix} \tag{4.14}$$

La matriz de covariancia es Hermitiana y positiva, por lo tanto, cada uno de los elementos de la matriz puede ser calculado como:

$$C_{xx}[j,k]=\frac{1}{2(N-p)}\left(\sum_{n=p}^{N-1} x^*[n-j]x[n-k]+\sum_{n=0}^{N-1-p} x[n+j]x^*[n+k]\right) \tag{4.15}$$

Basados en los parámetros del modelo obtenidos al resolver el sistema lineal, la estimación de

la variancia del ruido blanco es obtenida a partir de la siguiente ecuación:

$$\sigma^2=\hat{p}_{min}=C_{xx}[0,0] + \sum_{k=1}^{p} \hat{a}[k]C_{xx}[0,k] \qquad (4.16)$$

Finalmente, la estimación de la densidad espectral de potencia ($\mathbf{P_{AR}(f)}$) es obtenida usando la expresión:

$$\hat{P}_{AR}(f)=\frac{\hat{\sigma}^2}{|A(F_n)|^2}=\frac{\hat{\sigma}^2}{|1 + \hat{a}[1]\exp(-j2\pi f) + ... + \hat{a}[p]\exp(-j2\pi fp)|^2} \qquad (4.17)$$

El denominador de la ecuación 4.17 es evaluado en el rango de frecuencia normalizada [-1/2,1/2] en los puntos $f_n=(-1/2)+(n-1)/N$, donde n=1,...,N, y N es el número de muestras.

Cabe señalar, que para valores grandes de N y p, el cálculo de los parámetros es computacionalmente intensivo debido al cálculo de los elementos de la matriz de covariancia.

IV.4. IMPLEMENTACION USANDO ALGORITMOS GENETICOS.

Como se mencionó anteriormente, para encontrar los parámetros que minimicen la función de error y modelen el sistema usando el estimador espectral de covariancia modificada, se requiere calcular el valor de cada uno de los elementos que constituyen la matriz de covariancia y posteriormente, resolver el sistema de ecuaciones que se presenta. Las dimensiones de la matriz de covariancia y por tanto el orden del sistema de ecuaciones, está en función del orden del modelo del estimador, con lo cual, el tener un orden mayor del modelo implica un incremento en las dimensiones de la matriz presentando el algoritmo un costo computacional mayor.

Una alternativa para el cálculo de los parámetros del modelo es por medio del uso de *Algoritmos Genéticos*. Con el fin de determinar el conjunto de parámetros óptimos del filtro adaptable implicado en el método de estimación espectral de covariancia modificada, una población debe definirse inicialmente. Cada elemento de esta población representa una solución factible para este método paramétrico.

Como requerimos de un conjunto de parámetros, cada una de los cromosomas artificiales o cadenas representa el conjunto de coeficientes filtro adaptable [1][3]. Cada coeficiente decodificado como una subcadena binaria puede tomar valores entre *0* y $2^{L/p}$; donde *L* es la longitud de la cadena y *p* el orden del modelo. Por lo que, para una cadena de longitud 48 y un modelo de orden 6, hay 2^8 valores que cada parámetro puede tomar. Un procedimiento lineal es empleado para el mapeo de los enteros no signados decodificados para cada parámetro de $[0, 2^8]$

a $[A_{min}, A_{max}]$.

Cada uno de los elementos en un generación particular es evaluado para conocer su desempeño o valor de *fitness*. En general, esta evaluación se representa como un número positivo, por lo que se aplica una transformación que mapee la función objetivo a valores de desempeño o adaptabilidad (fitness). Esto es:

$$f(x) = \begin{cases} 1 - \dfrac{g(x)}{C_{max}} & g(x) \le C_{max} \\ 0 & de\ otro\ modo \end{cases} \tag{4.18}$$

donde: $f(x)$ es la función de desempeño (fitness).
C_{max} es el valor máximo de la función de error predictivo.

Un punto importante en la implementación genética es el orden del modelo, el cual corresponde al número de variables de decisión. Esto se ve reflejado en las dimensiones del espacio de soluciones. Como ejemplo podemos tener un modelo de orden 6 dando un espacio de 6 dimensiones, un espacio de soluciones grande y complejo. Pero de acuerdo al Teorema del Esquema [3][4], cada uno de los individuos de la población puede representar una región diferente de este espacio, con lo cual se están explorando simultáneamente diferentes regiones.

En el apéndice B se presenta la función objetivo a evaluar la cual corresponde a la función de error predictivo a minimizar del estimador de covariancia modificada, siendo ésta:

$$f(x) = \frac{1}{1(N-p)} \left(\sum_{n=p}^{N-1} \left| x[n] + \sum_{k=1}^{p} a[k]x[n-k] \right|^2 \right. \tag{4.19}$$

$$\left. + \sum_{n=0}^{N-1-p} \left| x[n] + \sum_{k=1}^{p} a^*[k]x[n+k] \right|^2 \right)$$

IV.5. CASO DE ESTUDIO.

Para comprobar el desempeño de la implementación genética de este estimador espectral, se presenta el análisis basado en una secuencia de 64 datos reales de un proceso consistente en tres señales senoidales con diferente frecuencia y una señal de ruido [7]. La Tabla 4.1. presenta una lista de los datos usados. La Densidad Espectral de Potencia Real se muestra en la gráfica

4.2. El eje de la frecuencia está en el rango de [0.0, 0.5] y representa la frecuencia de muestreo. Las tres señales senoidales están en las frecuencia 0.1, 0.2 y 0.21 Hz. La señal de ruido está centrada en 0.35 Hz.

Tabla I.

x[1] = 1.291061	x[33]= 0.309840
x[2] = -2.086368	x[34]= 1.212892
x[3] = -1.691316	x[35]= -0.119905
x[4] = 1.243138	x[36]= -0.441686
x[5] = 1.641072	x[37]= -0.879733
x[6] = -0.008688	x[38]= 0.306181
x[7] = -1.659390	x[39]= 0.795431
x[8] = -1.111467	x[40]= 0.189598
x[9] = 0.985908	x[41]= -0.342332
x[10]= 1.997919	x[42]= -0.328700
x[11]= -0.046613	x[43]= 0.197881
x[12]= -1.649269	x[44]= 0.071179
x[13]= -1.040810	x[45]= 0.185931
x[14]= 1.054665	x[46]= -0.324595
x[15]= 1.855816	x[47]= -0.366092
x[16]= -0.951182	x[48]= 0.368467
x[17]= -1.476495	x[49]= -0.191935
x[18]= -0.212242	x[50]= 0.519116
x[19]= 0.780202	x[51]= 0.008320
x[20]= 1.416003	x[52]= -0.425946
x[21]= 0.199202	x[53]= 0.651470
x[22]= -2.027026	x[54]= -0.639978
x[23]= -0.483577	x[55]= -0.344389
x[24]= 1.664913	x[56]= 0.814130
x[25]= 0.614114	x[57]= -0.385168
x[26]= -0.791469	x[58]= 0.064218
x[27]= -1.195311	x[59]= -0.380008
x[28]= 0.119801	x[60]= -0.163008
x[29]= 0.807635	x[61]= 1.180961
x[30]= 0.895236	x[62]= 0.114206
x[31]= -0.012734	x[63]= -0.667626
x[32]= -1.763842	x[64]= -0.814997

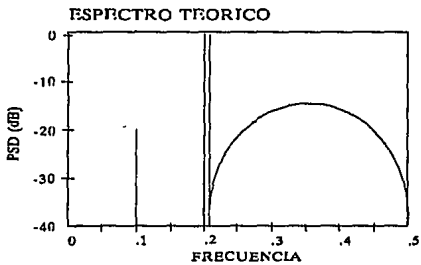


Figura 4.2. Espectro Teórico.

En la gráfica 4.3. se presentan los resultados arrojados al aplicar la Transformada Rápida de Fourier (FFT) y el método de estimación espectral de covariancia modificada con sus dos formas de solución: matriz de covariancia y algoritmos genéticos. El orden empleado es de p=6 y se puede observar que no se tiene una buena resolución en ambas formas de solución del método de covariancia modificada, ya que la señal senoidal localizada en la frecuencia de 0.1 Hz no está definida en estas respuestas espectrales. La respuesta espectral basada en el método de la Transformada Rápida de Fourier presenta señales espurias y un corrimiento en la señal senoidal localizada en la frecuencia de 0.1 Hz en el espectro teórico.

En la gráfica 4.4. el orden del modelo se incrementó teniendo p=10. Al incrementar el orden del modelo se incrementó la resolución de la respuesta en frecuencia comparada con la gráfica 4.2. Podemos observar que, tanto para la solución por medio de la matriz de covariancia como para la alternativa de Algoritmos Genéticos, la respuesta espectral define la señal senoidal de 0.1 Hz. Esta gráfica puede compararse con la correspondiente a la PSD real observando las frecuencias de las senoides, pero el punto principal es el hecho de comparar las respuestas de ambas formas de solución del estimador espectral con lo cual se puede ver un comportamiento similar en ambos casos. En el caso de la implementación por medio de algoritmos genéticos, en tan solo 5 generaciones se obtiene la respuesta presentada en la gráfica.

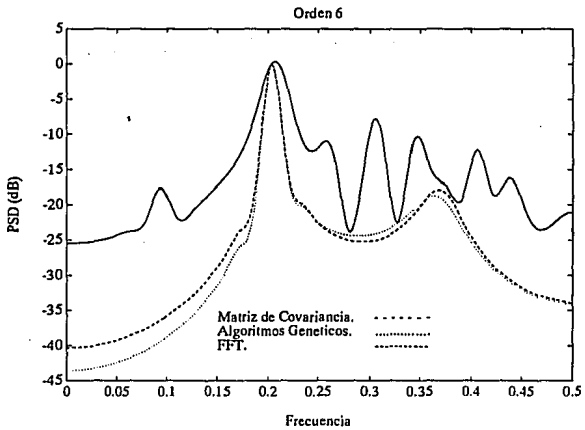


Figura 4.3. Covariancia Modificada Orden 6.

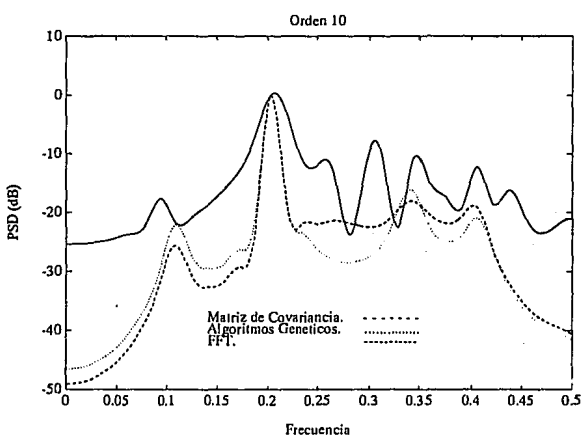


Figura 4.4. Covariancia Modificada Orden 10.

IV.6. CONCLUSIONES.

Los Algoritmos Genéticos son un método alternativo para el cálculo de los parámetros que forman la matriz de covariancia del modelo autorregresivo. Los resultados obtenidos muestran que con pocas generaciones se puede tener una buena estimación del espectro de la señal. El *paralelismo implícito* de los AG's permite al algoritmo explorar diferentes regiones del espacio de soluciones simultáneamente reduciendo el complejidad que implica la implementación del algoritmo. Esto permite la implementación de filtros de mayor orden, incrementando la resolución espectral, y abriendo la posibilidad de usar métodos de estimación espectral más complejos.

Con los resultados obtenidos, se presenta en el siguiente capítulo la estimación espectral aplicada en Flujometría Doppler.

IV.7. REFERENCIAS.

[1] Fleming, P.J., Chipperfield, A., Pohlheim, H., Fonseca, C., "Genetic Algorithms. Toolbox. User's Guide", University of Sheffield, 1994.

[2] García, D.F., Ruano, M.G., Fish, P.J., Fleming, P.J., "Parallel Implementation of a Model-Based Spectral Estimator for Doppler Blood Flow Instrumentation", 8th International Parallel Processing Symposium", IEEE, 1994.

[3] Goldberg, D.E., "Genetic Algorithms in Search, Optimization and Machine Learning", MA: Addison-Wesley, 1989.

[4] Holland, J.J., "Adaptation in Natural and Artificial Systems", Ann Arbor, MI: University of Michigan Press, 1975.

[5] Kay, S.M., "Modern Spectral Estimation", Prentice Hall, 1988.

[6] Marple, S., "Digital Spectral Analysis with Applications", Prentice Hall, 1987.

[7] Marple, S., Kay, S.M., "Spectrum Analysis: A Modern Perspective", Proceeding of IEEE, Vol. 69, 1981.

[8] Proakis, J.G., Manolakis, D.G., "Digital Signal Processing", Mcmillan Publishing, 1992.

[9] Ruano, M.G., "Investigation of Real-Time Spectral Analysis Techniques for Use with Pulsed Ultrasonic Doppler Blood-Flow Detectors", PhD Thesis, University of Wales, U.K., 1992.

CAPITULO V

ESTIMACION ESPECTRAL EN FLUJOMETRIA DOPPLER

V.1. INTRODUCCION.

En la actualidad una gran cantidad de decesos en la población son causados por enfermedades cardiovasculares. Dispositivos tales como el ***Detector de Flujo Sanguíneo por Efecto Doppler*** son usados con la finalidad de detectar y monitorear estos problemas.

Este instrumento determina la velocidad del torrente sanguíneo midiendo el desplazamiento Doppler en frecuencia del ultrasonido dispersado por el flujo sanguíneo. Un incremento en el rango de frecuencias Doppler, como resultado de algún tipo de turbulencia, es usado para detectar el grado de lesiones estenóticas.

Debido a que la velocidad de la sangre dentro de las arterias es periódica, la señal Doppler es ciclo estacionaria, variando su espectro en forma y frecuencia media durante del ciclo cardiaco. Esto hace necesario el tener que realizar el análisis espectral de la señal en intervalos de tiempo pequeños, típicamente (2-20 ms) [2], en los que la señal puede considerarse como estacionaria.

La gran mayoría de los sistemas de ultrasonido Doppler comercialmente disponibles hacen uso de la *Transformada Rápida de Fourier (FFT)* para calcular el espectro Doppler de la señal generada al monitorear el flujo sanguíneo, extrayendo de este modo importante información para su diagnóstico. De esta manera, padecimientos moderados y severos pueden ser detectados. Sin embargo, con esta técnica es difícil distinguir problemas de estenosis en su estado inicial, debido a las limitantes de la FFT asociadas a su resolución y segmentación.

Trabajos recientes de investigación han llegado a la conclusión de que los métodos *paramétricos de estimación espectral* ofrecen un importante avance en términos de resolución de la respuesta en frecuencia [7]. Otros estudios [5] realizados con el fin de investigar el desempeño de un número de estimadores espectrales en señales Doppler, han identificado el método paramétrico *autorregresivo* de ***Covariancia Modificada*** [3][4], como el más conveniente en términos de eficiencia y complejidad computacional, además de ser un método cuyas características permiten el uso de técnicas de procesamiento paralelo [6].

En el capítulo anterior se llevó a cabo un análisis comparativo de la respuesta en frecuencia obtenida utilizando la FFT y el método de Covariancia Modificada. Asimismo se

presentó un método alternativo basado en este estimador paramétrico implementado mediante el uso de ***Algoritmos Genéticos.*** Este método basado en conceptos de la evolución natural encuentra el conjunto de parámetros que minimizan la función de error del filtro adaptable utilizado en el método espectral.

Este capítulo presenta la implementación del estimador espectral paramétrico aplicado a ***Flujometría Doppler.*** El objetivo es poder reducir la complejidad computacional del estimador paramétrico utilizando la simplicidad inherente de los AG's, lo cual nos permitirá incrementar el orden del modelo del filtro generando una mejor resolución en la respuesta, además de que nos permitirá abrir la posibilidad de utilizar métodos aún más complejos.

V.2. ULTRASONIDO Y EFECTO DOPPLER.

V.2.1. Ultrasonido.

Una onda sónica o ultrasónica es provocada por un desorden mecánico de un medio (gas, líquido o sólido). Dichas ondas se generan por un desorden de moléculas, las vibraciones de las cuales pasan de molécula en molécula desde un origen a un destino. El rango en el cual las partículas vibran en el medio es conocido como frecuencia o grado del sonido y es medida en Hertz (ciclos/segundo). Ondas acústicas con frecuencias arriba de 20 KHz (rango no audible), son conocidas como ***ultrasonido.***

Existen dos tipos de movimiento de ondas: transversal y longitudinal. En las ondas transversales, el desorden es perpendicular a la dirección de propagación. Un ejemplo de ésto son las ondas generadas al pegar un objeto en la superficie del agua. Las ondas longitudinales poseen la propiedad de que el desorden o disturbio se efectúa en la misma dirección de la propagación de la onda. Ejemplo de éstas son el sonido y el ultrasonido. Estas ondas también son llamadas compresionales.

Durante la propagación del desorden o disturbio, las partículas se mueven muy cerca de sus partículas vecinas y comprimen el medio, ocasionando un incremento local en la presión. Cuando, como resultado de la elasticidad del medio, se mueven en dirección opuesta, crean una reducción en la presión local. Este cambio en la presión resultado del paso de un disturbio o desorden, es llamado presión excedente.

En ultrasonido médico [2], el desorden caracterizado por el cambio de presión local o distancia de movimiento de las partículas del medio desde su posición de reposo, se origina en un transductor electromecánico usualmente localizado sobre la superficie de la piel del paciente. El transductor, operando como transmisor, cambia las señales eléctricas en movimiento mecánico. Asimismo, las vibraciones mecánicas pueden ser cambiadas a señales eléctricas por el mismo transductor trabajando en modo inverso (como receptor). Típicamente, las frecuencias de ultrasonido usadas son altas (en la región de 1-20 MHz).

V.2.2. Efecto Doppler.

El efecto doppler es un corrimiento en la frecuencia observada de una onda radiada cuando hay un movimiento relativo entre la fuente de radiación y un observador [2].

Pueden existir tres casos para analizar el efecto doppler: fuente en movimiento, observador en movimiento y reflector en movimiento.

Fuente en Movimiento.

Basados en la figura 5.1.a., si tanto la fuente como el observador son estacionarios, las crestas de la ondas, emitidas en un rango regular (frecuencia de transmisión), viajan a través del medio a una velocidad constante, cada cresta separada de la siguiente una cierta distancia, y pasa por el observador al mismo rango transmitido. Si la fuente se mueve hacia el observador (figura 5.1.b.), significa que el espaciamiento entre crestas será más pequeño y por lo tanto, la longitud de onda se reduce. Estas crestas pasan por el observador a la misma velocidad experimentando un incremento en la frecuencia de las ondas. Por el contrario, si la fuente se aleja del observador, la longitud de onda se incrementa y el observador experimenta una reducción en la frecuencia de las ondas.

Observador en Movimiento.

Si la fuente es estacionaria y el observador está moviéndose hacia la fuente (figura 5.2.c.), el observador intercepta las crestas a una mayor velocidad experimentando un incremento en la frecuencia. De manera inversa, si el observador se aleja de la fuente, el observador intercepta las crestas a una menor velocidad disminuyendo con ésto la frecuencia. Para el caso que tanto la fuente como el observador presenten movimiento, la diferencia entre las frecuencias recibidas y transmitidas es conocida como *Corrimiento Doppler*.

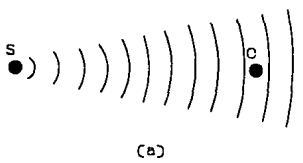


Figura 5.1.a.

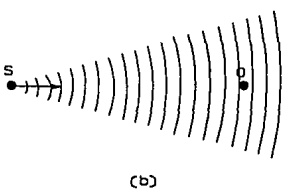
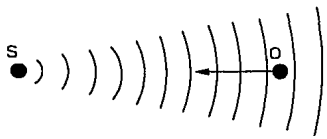


Figura 5.1.b.



(c)

Figura 5.1.c.

Reflector en Movimiento.

En los instrumento médicos ultrasónicos Doppler tanto la fuente (el transductor de transmisión) como el observador (transductor receptor) están montados en una sonda y son estacionarios con respecto al medio (tejido). Algún reflector o dispersor de ultrasonido puede ser considerado primero como observador del ultrasonido transmitido y después como fuente de ultrasonido para el receptor. Si el dispersor o reflector es estacionario, el receptor presenta la misma frecuencia que la transmitida (figura 5.2.a.). Si el reflector se mueve desde la sonda ultrasónica (figura 5.2.b.) entonces, primero como observador del ultrasonido transmitido presentará una frecuencia más baja que la transmitida, y después como fuente para el transductor de recepción emitirá una señal ultrasónica de una longitud de onda mayor que la que ha sido observada. Para el caso del reflector en movimiento se presentan dos desplazamientos Doppler en la misma dirección experimentando el observador una frecuencia más baja que la transmitida. Por el contrario, si el dispersor se mueve hacia la sonda, el observador presenta una frecuencia más alta que la transmitida. Por tanto, el corrimiento Doppler (la diferencia entre la frecuencia observada y la transmitida) es proporcional a la velocidad del dispersor. En ultrasonido Doppler médico es frecuente el movimiento en un ángulo de las señales de transmisión y recepción como se muestra en la figura 5.3. La frecuencia de corrimiento Doppler está entonces dada por:

$$f_d = \frac{-2Vf_o\cos(\theta)\cos(\delta/2)}{c} \qquad (5.1)$$

donde:

V = velocidad del reflector o dispersor.
c = velocidad de ultrasonido.
Θ = ángulo entre el bisector de la señal transmitida y recibida y la dirección de movimiento.
δ = ángulo entre la señal transmitida y la recibida.
f_o= frecuencia transmitida.

El signo negativo indica que el corrimiento Doppler es negativo (frecuencia transmitida desplazada a frecuencias más bajas) si la dirección del movimiento es la dirección positiva convencional (hacia los transductores).

En los instrumentos Doppler de tipo pulsado, el mismo transductor es usado en la transmisión y recepción, el ángulo (δ) entre las señales es muy pequeño y $cos(\delta/2)$ tiende a 1.0. De este modo, la ecuación 5.1 puede simplificarse a:

$$f_d = \frac{-2Vf_o\cos(\theta)}{C} \qquad (5.2)$$

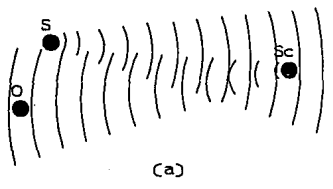


Figura 5.2.a.

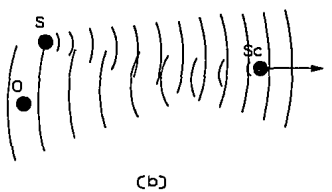


Figura 5.2.b.

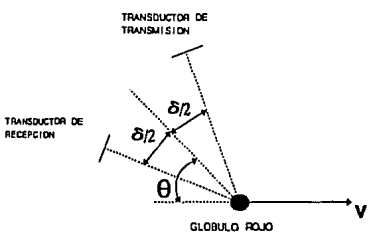


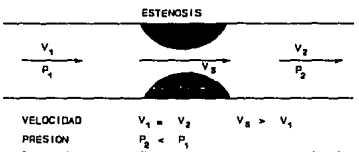
Figura 5.3.

Por otra parte, existen dos tipos de instrumentos Doppler: el instrumento Doppler de Onda Continua y el instrumento Doppler Pulsado. En los instrumentos Doppler de Onda Continua se tienen dos transductores (uno de recepción y otro de transmisión). Esto presenta la limitación de sensibilidad al flujo en la región de traslape de los dos rayos ultrasónicos. Manejar un solo rayo evita el problema de tener una región muy particular de prueba dependiendo del ángulo manejado por los dos transductores. Este sistema está integrado en el instrumento Doppler Pulsado.

V.3. ANALISIS ESPECTRAL DE SEÑALES DOPPLER.

El *análisis espectral* puede ser usado para detectar la presencia de lesiones estenóticas al indicar el incremento de la velocidad en el flujo a través de la estenosis, junto con el incremento en el rango de las frecuencias de corrimiento Doppler. La figura 5.4. muestra una lesión estenótica, la cual afecta la distribución de velocidad y presión del flujo sanguíneo, ocasionando turbulencias en el mismo.

La estenosis se forma preferentemente, pero no exclusivamente, en las bifurcaciones de los vasos sanguíneos. Como ejemplo se tiene la bifurcación de la carótida común a las arterias carótidas interna y externa.



Figra 5.4. Lesión Estenótica.

Al agudizarse una lesión estenótica (decremento en el diámetro de los vasos sanguíneos) se genera una elevación progresiva en la presión a través de la estenosis, un decremento en el flujo sanguíneo y un incremento en la velocidad, como se muestra en la figura 5.5. Es importante mencionar que es necesario tener un alto grado de la lesión (reducción importante en el diámetro de los vasos sanguíneos) antes que ésta se refleje en una caída significativa en el rango del flujo sanguíneo. Cuando el tamaño de un vaso sanguíneo se decrementa, la velocidad crece a un punto máximo, alcanzando el flujo rápidamente el cero, significando con ésto una oclusión completa.

La velocidad en la estenosis se mantiene constante hasta el momento en que se presenta una lesión aguda. Debido a la naturaleza pulsátil del flujo sanguíneo en las arterias, como resultado de la acción de bombeo del corazón, una lesión estenótica afecta la forma de onda de la velocidad de flujo como se aprecia en la figura 5.6.

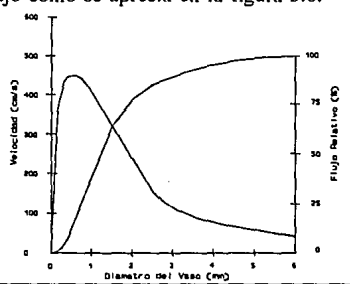


Figura 5.5. Presión, velocidad y flujo en una estenosis.

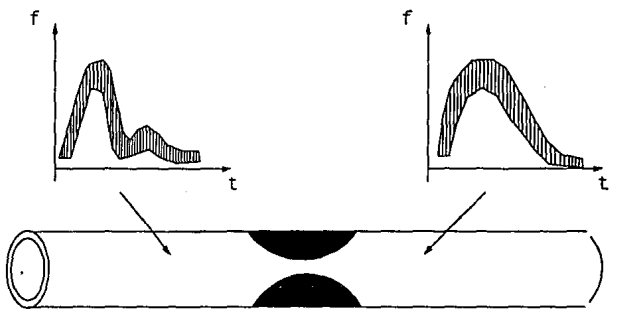


Figura 5.6. Velocidad del Flujo en las Arterias.

Cuando la estenosis no es lo suficientemente severa como para causar una reducción significativa en el flujo sanguíneo, y por lo tanto reducir el transporte de oxígeno y nutrientes a los tejidos, la lesión puede aun tener un efecto severo por las partículas arrastradas bajo la acción del flujo pulsátil y la flexión periódica de las paredes de los vasos (figura 5.7.). Estas partículas pueden ser lo suficientemente grandes para obstruir o bloquear vasos pequeños.

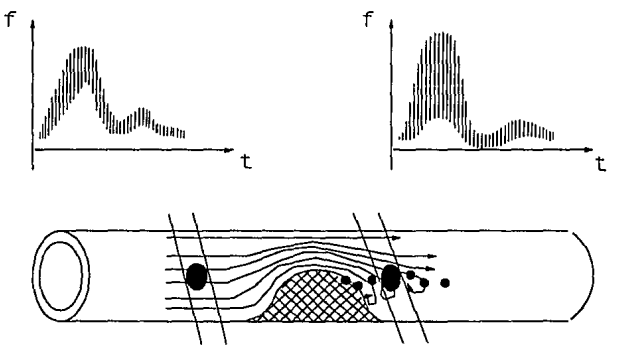


Figura 5.7. Flujo Turbulento.

Desde el momento en que la señal Doppler se genera por el paso de una colección aleatoria de células de sangre a través del volumen de muestra, la señal Doppler misma posee una naturaleza aleatoria. Como el espectro en diferentes tiempos del ciclo cardiaco es estimado de segmentos cortos de tiempo (2-20 ms) de la señal Doppler, el espectro medido para un ciclo cardiaco es muy irregular (figura 5.8). Con el fin de realizar una medición del ancho espectral para ayudar a la discriminación entre diferentes grados de estenosis, el espectro medido en diferentes puntos en un número de ciclos cardiacos puede obtenerse como un promedio de dichos espectros. Esto es realizado con la finalidad de obtener una medición más exacta del ancho del espectro.

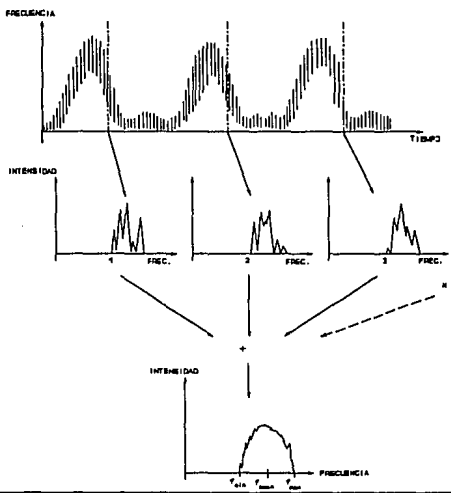


Figura 5.8. Esspectro en Frecuencia Promedio.

V.3.1. Estimación Espectral Paramétrica.

Como se mencionó anteriormente, el análisis espectral es una herramienta útil en Medicina para la detección y diagnóstico de ciertas afecciones. Para este análisis espectral tradicionalmente se han utilizado diferentes métodos de estimación. Los métodos de estimación espectral convencionales emplean la *Transformada Rápida de Fourier (FFT)* sobre segmentos de datos o ventanas. Estos métodos presentan una inadecuada resolución en frecuencia debido a la duración del segmento y a las características no estacionarias de las señales Doppler.

Otros métodos tales como los *estimadores paramétricos*, tambiés conocidos como *estimadores basados en modelos*, buscan igualar la función de autocorrelación de la señal a

analizar con una señal de ruido filtrado. Dicha igualación se logra alterando los parámetros del filtro, dando como resultado mejoras significativas en la resolución de la respuesta en frecuencia de la señal.

Existen diversos métodos de estimación paramétrica que difieren en: el modelo del filtro, el procedimiento para igualar la salida del filtro con la señal real considerada y el método utilizado para encontrar los parámetros asociados al filtro. Diversos estudios han identificado el método de ***Covariancia Modificada*** como el más adecuado en términos de *costo/beneficio* para el tipo de señales de nuestro interés, y por tanto, es el método considerado en este trabajo.

VI.4. IMPLEMENTACION Y ANALISIS.

Con la finalidad de evaluar la implementación genética del estimador espectral en Flujometría Doppler con respecto a los resultados encontrados a través de la solución del sistema de ecuaciones de la matriz de covariancia, ambos métodos utilizaron una secuencia de segmentos de una señal Doppler simulada, cada una con un número de datos de $N=256$ determinado por el producto de la frecuencia de muestreo y la duración de la secuencia de datos. Para este trabajo, la duración del segmento ha sido fijada en *20 ms.*, y, tanto la frecuencia de muestreo como la longitud de la secuencia de datos han sido ajustadas para satisfacer las restricciones de la señal.

Para el caso genético, el procedimiento de selección *Ranking* ha sido elegido basados en el estudio realizado en el capítulo III. Una población inicial de $popsize=30$, con $P_c=1.0$ y $P_m=0.01$ han sido consideradas para determinar los parámetros óptimos del filtro adaptable de orden $p=6$ para el modelo del estimador. Asimismo, una cadena de longitud $L=42$ es empleada, la cual proporciona 2^7 posibles valores para cada parámetro.

Los resultados comparativos se muestran en la figura 5.9. La gráfica superior izquierda muestra la densidad espectral de potencia (PSD) calculada por el método de covariancia modificada, encontrando los parámetros del filtro por medio de la solución del *sistema de ecuaciones de la matriz de covariancia.* Las gráficas restantes corresponden al método de covariancia modificada pero esta vez usando la implementación genética con *7 bits* de resolución para cada parámetro. La gráfica superior derecha representa la PSD la cual es obtenida con los valores de los parámetros dados después de una generación. Se observa que el espectro difiere del calculado por la solución de la matriz de covariancia. No obstante, después de dos generaciones (gráfica inferior izquierda para 2 generaciones y la inferior derecha para 3 generaciones), la respuesta en frecuencia es muy semejante difiriendo solo en magnitud.

Una variación de los parámetros del AG es llevada a cabo cambiando la longitud de cada miembro de la población a $L= 48$, manteniendo los mismos valores para los demás parámetros. De esta manera, cada uno de los parámetros del filtro pueden ahora tomar 2^8 valores diferentes, incrementando así su resolución.

Se puede observar en la figura 5.10. que, el AG mejora la respuesta espectral (PSD) en pocas generaciones y para la generación 3 (gráfica inferior derecha), se obtiene una respuesta espectral igual en frecuencia y magnitud a la presentada por el sistema de ecuaciones de la matriz de covariancia.

En la figura 5.11. se concentran los resultados de las tres implementaciones contempladas. En la gráfica superior izquierda se presenta una ventana de 20 ms de la señal Doppler simulada a la cual se aplicó el análisis espectral. En la gráfica inferior izquierda se tiene la PSD de la señal aplicando el método de la Transformada Rápida de Fourier, en la cual se observa su inadecuada resolución en frecuencia. Las dos gráficas inferior y superior derecha corresponden al método de covariancia modificada empleando la matriz de covariancia y AG's (8 bits de resolución), respectivamente.

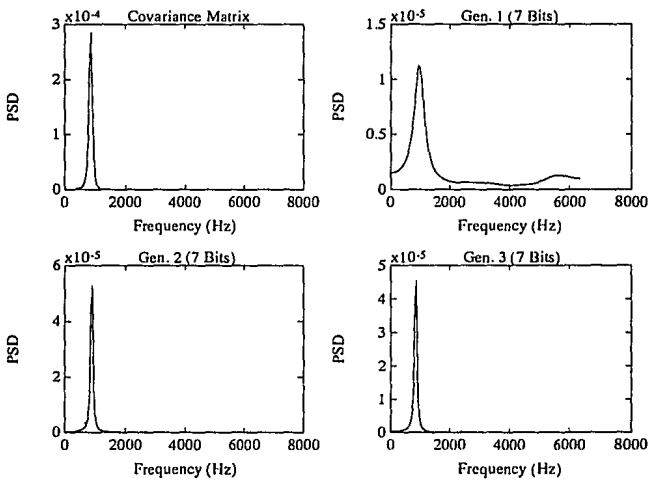


Figura 5.9. Solución Genética (7 bits de resolución).

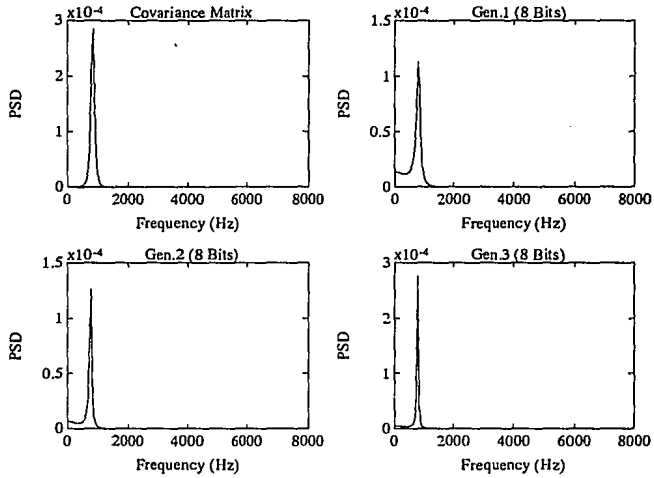


Figura 5.10. Solución Genética (8 bits de resolución).

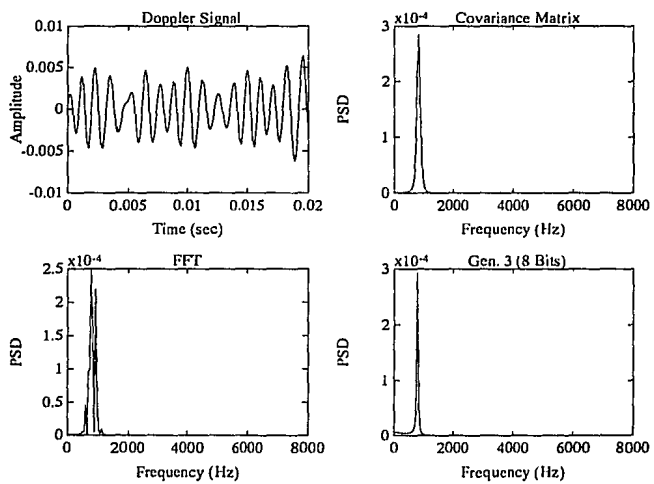


Figura 5.11. Resultados Comparativos.

V.5. CONCLUSIONES.

Varias aproximaciones han sido usadas en análisis espectral de señales Doppler. Los métodos convencionales que hacen uso de la FFT, sufren de una resolución en frecuencia inadecuada debida a la duración del segmento y las características no estacionarias de estas señales. Los métodos paramétricos pueden presentar un mejoramiento significativo en la resolución en términos de frecuencia. Sin embargo, cuando el número de datos y el orden del modelo del estimador paramétrico es alto, el cálculo de los parámetros del filtro al resolver la matriz de covariancia puede ser un proceso que consuma mucho tiempo.

El trabajo presentado en estos capítulos, muestran una alternativa para encontrar los parámetros óptimos del modelo del estimador basados en la simplicidad asociada a los Algoritmos Genéticos.

Los resultados obtenidos muestran que en pocas generaciones podemos tener una buena estimación del espectro de la señal Doppler con el uso de AG's. Observando las respuestas espectrales de los dos métodos utilizados tenemos respuestas muy semejantes, lo cual nos demuestra el buen desempeño de la implementación genética.

La implementación genética realizada en este capítulo, ha sido llevada a cabo mediante el uso de un Algoritmo Genético Secuencial ejecutado en un transputer T800.

No obstante los resultados obtenidos, los AG's presentan atractivas ventajas para explotar su naturaleza paralela, y así poder reducir significativamente el tiempo de procesamiento del algoritmo.

El capítulo VI plantea esta alternativa, analizando el desempeño de diferentes modelos de paralelismo aplicados al problema de estimación espectral paramétrica.

V.6. REFERENCIAS.

[1] Fish, P.J., "The Pulsed Ultrasonic Blood-Flow Detector. Factors Affecting Measured Doppler Signal Spectrum-Width in Undisturbed Flow", Proceeding 2nd Portuguese Congress on Biomedical Engineering, Aveiro, Portugal, 1990.

[2] Fish, P.J., "Physics and Instrumentation of Diagnostic Medical Ultrasound", Chichester: Wiley, 1990.

[3] Kay, S.M., "Modern Spectral Estimation", Prentice Hall, 1990.

[4] Marple, S.L., Kay, S.M., "Spectrum Analysis: A Modern Perspective", Proceedings of the IEEE, Vol 69, pp. 1380-1419, 1981.

[5] Ruano, M.G., "Investigation of Real-Time Spectral Analysis Techniques for Use with Pulsed Ultrasonic Doppler Blood-Flow Detectors", PhD Thesis, University College of

North Wales, Bangor, UK., 1992.

[6] Ruano, M.G., García, F., Fish, P.J., Flemming, P.J., "Parallel Implementation of s Model-Based Spectral Estimator for Doppler Blood-Flow Instrumentation", 8th International Parallel Processing Symposium, IEEE, 1994.

[7] Vaitkus, P., Cobbold, R., "A Comparative Study ans Assessment of Doppler Ultrasound Spectral Estimation Techniques", Ultrasound in Medicine and Biology, 1988.

CAPITULO VI

MODELOS DE PARALELISMO EN ALGORITMOS GENETICOS

La implementación de un Estimador Espectral Paramétrico empleando Algoritmos Genéticos fue desarrollado en el capítulo IV, presentándose en el capítulo V el análisis espectral de una señal Doppler mediante las dos formas de solución consideradas para el estimador paramétrico: el método de covariancia modificada y la alternativa de Algoritmos Genéticos. Esta sección del trabajo presenta la implementación de diferentes modelos de paralelismo de este Algoritmo Genético tratando de explotar la naturaleza paralela de los mismos, y teniendo como objetivo el estudio del desempeño de estos sistemas de procesamiento paralelo con las ventajas asociadas que dichos modelos pueden ofrecer en relación a la versión secuencial del Algoritmo Genético.

VI.1. INTRODUCCION.

En los últimos años y motivados por el constante desarrollo de arquitecturas en paralelo y la multiprogramación, muchas investigaciones dentro del campo de los Algoritmos Genéticos se han dirigido hacia el desarrollo de versiones de *Algoritmos Genéticos Paralelos (AGP's)* [12].

Desde los trabajos iniciales de **Holland** [7] en el campo de los Algoritmos Genéticos, se han dado muchas variaciones de los mismos. El desarrollo de Algoritmos Genéticos se ha fijado el objetivo de mantener la diversidad de las muestras y la eficiencia.

Con respecto al paralelismo, han habido dos áreas de desarrollo más allá de los Algoritmos Genéticos propuestos por Holland. La primera de ellas abarca los ***Algoritmos Genéticos Paralelos de Granularidad Media o Grande***, en los cuales varias poblaciones grandes evolucionan en paralelo con muy pocas interacciones. La segunda de ellas incluye los ***Algoritmos Genéticos Paralelos de Granularidad Fina***, en los cuales numerosas poblaciones pequeñas están constantemente interactuando y evolucionando en paralelo. Además, con la disponibilidad que se tiene actualmente de hardware más poderoso, estos métodos de optimización pueden ahora ser aplicados a problemas más complejos abarcando un espacio mayor de soluciones.

De manera general, los ***Algoritmos Genéticos Paralelos (AGP's)*** están constituidos por un conjunto de *Algoritmos Genéticos Nodales* [10], cada uno de los cuales está residente en un nodo de un sistema multiprocesador, manteniendo una parte de la población al ser dividida dicha población en subpoblaciones correspondientes al número de procesadores disponibles. Cada uno de los nodos ejecuta el mismo algoritmo independientemente o con interacciones con otras subpoblaciones. El *Algoritmo Genético Nodal* básico puede escribirse de la siguiente manera:

```
AG nodal:
        Inicialización
        Evaluación
        Mientras se ejecute
                Comunicación y Selección
                Reproducción
                Mutación
                Evaluación
                Substitución
        Fin
Fin
```

Excepto por la fase de comunicación, la cual puede ocurrir antes o después del proceso de selección, un Algoritmo Genético Nodal es idéntico a un Algoritmo Genético Secuencial. Durante la fase de comunicación, una subpoblación intercambia información con otra residente en un nodo diferente. Este proceso de comunicación puede realizarse entre dos subpoblaciones (o nodos) arbitrarios.

VI.2. RAZONES DEL PARALELISMO.

Como se mencionó anteriormente, una población genética es dividida en subpoblaciones para su paralelización. Una de las razones de esta división es el incremento potencial en la velocidad de ejecución por la asignación de subpoblaciones a cada procesador del sistema. Sin embargo, una razón más importante se deriva de la observación que, después de ciertas evoluciones, la mayoría de los cromosomas artificales en la población, presentarán un comportamiento similar. La diversidad genética se perderá, y la recombinación después de ésto puede no ser productiva. Un método para atacar este problema es evolucionar las subpoblaciones independientemente.

Como los Algoritmos Genéticos son métodos aleatorios de búsqueda, las evoluciones independientes estarán explorando diferentes regiones del espacio de soluciones. Si las funciones a ser optimizadas en cada uno de los AG nodales son las mismas, cada subpoblación debe ser muy competitiva, y aun más, presentar resultados únicos.

El número de interacciones entre subpoblaciones puede ser un factor crítico en la determinación de la eficiencia de un Algoritmo Genético Paralelo. La eliminación de interacciones entre subpoblaciones hace que una población grande presente un desempeño similar al correr varios AG's con poblaciones pequeñas, mientras que, con un número grande de interacciones, los beneficios de las subpoblaciones se pierden. Los mejores individuos o cromosomas de una subpoblación se propagan rápidamente a otras subpoblaciones, y las evoluciones ya no permanecen independientes.

VI.3. IMPLEMENTACION PARALELA USANDO UNA ARQUITECTURA DE PROCESO DE COMUNICACION.

Por ***Arquitectura de Proceso de Comunicación (APC)*** se entiende un sistema compuesto de procesos los cuales se comunican síncronamente por medio de canales de comunicación de acuerdo a protocolos. El paradigma de ***Procesos Secuenciales de Comunicación (PSC)*** [6] fue desarrollado por *Hoare* durante los años 70's. Está basado en la concurrencia y comunicación de procesos.

Mientras que dicho paradigma es una base matemática concerniente con el análisis de sistemas, una APC busca determinar la manera de cómo estas mismas ideas pueden ser aplicadas a la ingeniería de sistemas paralelos. Una de las herramientas disponibles para la construcción de un sistema APC es el compilador para el lenguaje de programación OCCAM [4][8], el cual, como se describió en el capítulo II, es un lenguaje muy sencillo. Está diseñado para expresar el funcionamiento de un sistema APC, incluyendo declaraciones de canales de entrada, un constructor para expresar la ejecución concurrente de una lista de procesos (*PAR*), y un constructor para la selección de un proceso de una lista de acuerdo a la disponibilidad de comunicación que presente (*ALT*).

Los requerimientos fundamentales de un sistema CPA son:

Validez. Por validez se entiende que el sistema encontrará su especificación en contra del aplazamiento de alguna salida en respuesta a una cierta entrada. Quizás, el problema más grave que puede ocurrir en un sistema APC es el **deadlock** de comunicación.

Eficiencia. La eficiencia [9] es la máxima utilización de los recursos de procesamiento y comunicación. La eficiencia en el procesamiento paralelo es usualmente definida como:

$$E_N = \frac{T_1}{N \cdot T_N} \qquad (6.1)$$

donde T_1 es el tiempo que toma un solo procesador en realizar una tarea, y T_N es el tiempo que toman N procesadores en finalizar la misma tarea.

Escalabilidad. Por escalabilidad se entiende que el software debe ser ejecutable, conservando la eficiencia, sobre una red grande de la misma topología con solo una recompilación. Esto puede ser llamado *escalabilidad de implementación.* Otra variedad de escalabilidad es la denominada *escalabilidad de aplicación*, que se entiende como la conservación de la eficiencia a medida que la escala del problema (datos de entrada) se incrementa.

La *escalabilidad* es uno de los puntos importantes en el procesamiento paralelo. Inicialmente, una topología escalable debe encontrarse con el fin de resolver el problema de interconexión escalable. Porteriormente, la configuración de procesos a procesadores debe especificarse de una manera escalable. Finalmente, el algoritmo debe escalarse de manera tal que la eficiencia se conserve.

VI.3.1. Comunicación.

El análisis de la forma de comunicación requerida puede darnos una mejor implementación del diseño del sistema paralelo. En un algoritmo **APC** se pueden presentar tres tipos de comunicación. Esto es:

Nodo a Host. Este tipo de comunicación es generalmente requerido para accesar los servicios provenientes del *host.* Sin embargo, puede no ser necesariamente el caso que todos los nodos requieran acceso al *host.* El acceso al *host* en un **AGP** es requerido para la inicialización del sistema, monitoreo y comunicación de resultados.

Vecindad. Este tipo de comunicación se refiere al intercambio de datos entre procesos vecinos.

Nodo a nodo. La comunicación nodo a nodo se refiere a la vía de comunicación entre un par cualquiera de nodos. Sin embargo, si el procesamiento local requiere el acceso a un conjunto de datos el cual es demasiado grande como para ser acomodado localmente, entonces se debe elegir entre un sistema de memoria distribuída y memoria virtual soportado centralmente (ej., uso de comunicación nodo a host y nodo a nodo).

VI.4. MODELOS PARA LA PARALELIZACION DE ALGORITMOS GENETICOS.

El interés en los *Algoritmos Genéticos Paralelos* ha ido creciendo a pasos agigantados con la disponibilidad comercial de computadoras paralelas. Las razones para ésto son fáciles de

adivinar: la mayoría de los métodos de búsqueda tradicionales están diseñados para máquinas secuenciales que hacen su paralelización eficiente un tanto difícil. Los Algoritmos Genéticos se comportan en forma totalmente diferente. Además de los conceptos convencionales para la paralelización donde el cálculo de subtareas independientes es difundido en varios procesadores, los Algoritmos Genéticos pueden ser conceptualmente ampliados para hacerlos correr en computadoras paralelas.

En contraste con las estrategias convencionales, la paralelización de los Algoritmos Genéticos no solo tiene como fin acelerar el proceso de cálculo, sino también generar soluciones con una mayor calidad.

Nuevamente podemos observar que la naturaleza es el ejemplo de estos esquemas de paralelización. Una cierta especie en el medio natural, por ejemplo, no está distribuída en una población grande y *panmíctica* [1][5], en la cual individuos pueden ser recombinados arbitrariamente. Más que una especie es una dispersión en subpoblaciones, cada una de las cuales puede evolucionar independientemente de otras subpoblaciones. Solo en intervalos irregulares algunos individuos pueden moverse a una nueva subpoblación.

Refiriéndonos a los sistemas multiprocesadores, cada subpoblación puede establecerse en un procesador y trabajar un Algoritmo Genético Secuencial convencional. Las subpoblaciones aisladas están relacionadas con diferentes procesadores. En general, el intercambio de individuos es hecho con un vecino el cual tiene que ser definido entre los procesadores.

Con la información anterior podemos ahora describir tres modelos diferentes para la paralelización de los Algoritmos Genéticos. Estos son el *Modelo Farming*, el *Modelo de Migración*, y el *Modelo de Difusión*.

VI.4.1. Modelo Farming.

En general, el modelo **Farming** cuenta con un proceso *maestro* y con **N** procesos *esclavos*. El proceso maestro distribuye las tareas a realizar en todos los procesos esclavos y posteriormente recibe los resultados de los esclavos.

En los Algoritmos Genéticos Paralelos basados en este modelo [1][11] se tiene un procesador *maestro* el cual supervisa la población genética total y realiza la selección. Los procesadores *esclavos* reciben los individuos que van a ser recombinados para crear los miembros de la nueva población. Estos nuevos individuos son evaluados antes de regresar al procesador maestro.

Comparando con la definición del Algoritmo Genético Nodal, en el modelo Farming los

procesos *Maestro* y *Esclavo* difieren de dicha definición. Estos dos nuevos algoritmos podemos escribirlos de la siguiente manera:

```
Algoritmo Genético Maestro:

        Inicialización
        Evaluación
        Mientras se ejecute
                Selección
                Comunicación Maestro-Esclavos
                Comunicación Esclavos-Maestro
                Substitución
        Fin
Fin
```

Algoritmo Genético Maestro

```
Algoritmo Genético Esclavo:

        Mientras se ejecute
                Comunicación Maestro-Esclavo
                Recombinación
                Mutación
                Evaluación
                Comunicación Esclavo-Maestro
        Fin
Fin
```

Algoritmo Genético Esclavo

Farming se aplica a tareas donde un número de tareas independientes de trabajo pueden llevarse a cabo simultáneamente. El proceso inicial es dividido en **paquetes de trabajo** por un **proceso controlador** (proceso maestro) el cual envía estos paquetes a **procesos de trabajo** (proceso esclavo) idénticos. Farming puede implementarse directamente con Algoritmos Genéticos Paralelos (AGP's) en los cuales las recombinaciones, mutaciones y evaluaciones de los individuos pueden realizarse en forma independiente.

De manera general, se tendrá un proceso esclavo por procesador. Sin embargo, el conectar el proceso controlador directamente a todos los esclavos no es factible, por lo cual es necesario conectar los procesadores mediante una topología adecuada. Para tal efecto pueden utilizarse dos topología ***pipeline*** y ***árbol*** (Figuras 6.1 y 6.2, respectivamente), adicionando un proceso de ruteo de envío de paquetes a los procesadores esclavos y recepción de resultados por parte del procesador maestro.

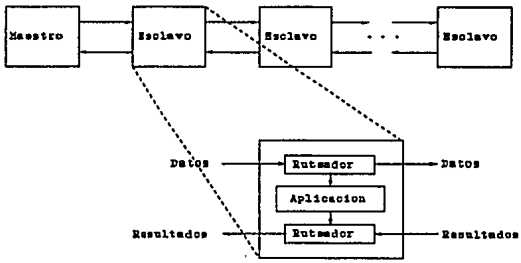


Figura 6.1. Topología Pipeline.

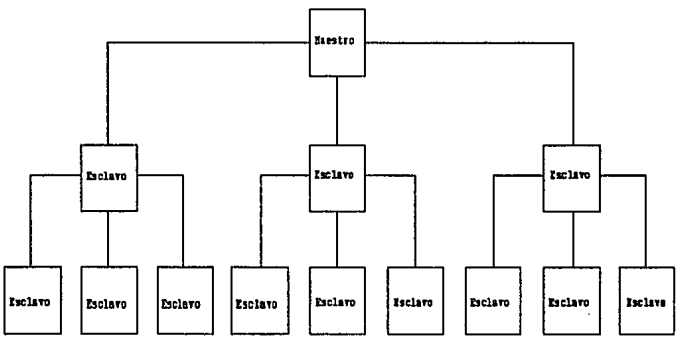


Figura 6.2. Topología Arbol.

VI.4.2. Modelo de Migración.

El conjunto de individuos con los cuales un miembro de una población puede cruzarse es conocido como *deme* [1]. Si el *deme* para todos los miembros de una población es igual a la población completa, entonces se dice que esa población es ***panmíctica***. Como ejemplo se tiene un Algoritmo Genético Simple o Secuencial, en el cual cualquier individuo puede cruzarse con algún otro.

En el medio natural, las poblaciones están frecuentemente **distribuidas**. El ejemplo más sencillo de un **AGP** con población distribuída es modelado en una red de "islas", cada una de las cuales, en aislamiento, es panmíctica.

De lo anterior se deriva el modelo de migración para los AGP's [1][3][11][13]. En este modelo, cada procesador contiene una subpoblación residente corriendo un **Algoritmo Genético Nodal**. En intervalos arbitrarios pueden moverse individuos de una subpoblación a otra. Este movimiento puede significar una ***emigración*** , cuando un individuo deja la subpoblación para integrarse a otra, o una ***inmigración*** al recibir una subpoblación a un individuo proveniente de una subpoblación vecina. Este modelo puede describirse de la siguiente manera:

```
Cada subpoblación paralelamente:
        Inicialización
        Evaluación
        Mientras se ejecute
                Selección
                Reproducción
                Mutación
                Evaluación
                Substitución
                PAR
                        Envía emigrantes
                        Recibe inmigrantes
        Fin
Fin
```

AGP Modelo de Migración

Es importante hacer notar que los inmigrantes reemplazan directamente a los emigrantes y son iguales en número (el tamaño de la población se conserva).

Una propiedad importante de este modelo de migración es que el envío y recepción de migrantes es realizado en paralelo, con lo cual tenemos la sincronización de los procesos.

VI.4.3. Modelo de Difusión.

A diferencia del modelo de migración, en el modelo de difusión [1][2][11] las subpoblaciones no están aisladas completamente, sino que se tiene un acceso aleatorio a todos los individuos en algún tiempo, los cuales están colocados dentro de una vencidad. Todos estos individuos pueden estar en continuo movimiento dentro de su vencidad, lo cual permite el acceso para la recombinación. Las vencindades parcialmente traslapadas soportan la difusión de material genético a diferentes subpobaciones. Como el grado de aislamiento de las subpoblaciones es

determinado por el grado de su traslape, en la genética de las poblaciones el término **aislamiento por distancia** es usado para caracterizar este modelo.

Un AGP basado en este modelo puede describirse de la siguiente manera:

```
Para cada nodo I(i,j)
      Inicialización
      Evaluación
      Mientras se ejecute
            PAR
              Envía emigrantes a subpobs. vecinas
              Recibe inmigrantes de subpobs. vecinas
            Selección
            Reproducción
            Mutación
            Evaluación
            Substitución
      Fin
Fin
```

AGP Modelo de Difusión

La topología propuesta para este modelo es un *toroide* presentada en la figura 6.3.

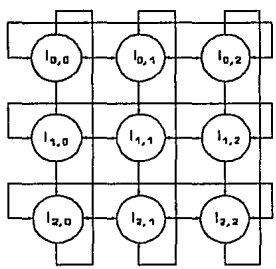


Figura 6.3. Topología Tipo Toroidal.

VI.5. IMPLEMENTACION DE MODELOS DE AGP's.

Las implementaciones de estos tres modelos de Algoritmos Genéticos Paralelos han sido realizadas en el lenguaje de programación **OCCAM** para correr sobre una plataforma de **Transputers**. En el capítulo II se presentó la definición y características del Transputer, además de una breve introducción del lenguaje OCCAM. En el apéndice C se presenta información complementaria de ésto, en la cual tenemos el modelo de programación de OCCAM, los pasos a seguir para correr un programa realizado en OCCAM sobre un transputer, y la manera de configurar una red de estos procesadores para correr un algoritmo paralelo.

La implementación de los modelos de AGP's descritos están basados en el problema de Estimación Espectral Paramétrica de Covariancia Modificada. Esto con la finalidad de poder tener un marco comparativo de eficiencia entre un Algoritmo Genético Secuencial (basados en los resultados del capítulo V) y los diferentes modelos de AGP's.

VI.5.1. Algoritmo Genético Paralelo Modelo Farming.

Para la implementación de este modelo se crearon dos Algoritmos Genéticos a partir del Algoritmo Genético Secuencial convencional, un Algoritmo **Maestro** y un Algoritmo **Esclavo.**

El Algoritmo **Maestro** realiza los procesos de creación de la población inicial, evaluación de los individuos generados aleatoriamente y el proceso de selección. En esta fase de selección, el algoritmo maestro crea un vector ***'matepool'*** el cual contiene todas las parejas de individuos que serán recombinadas y mutadas para la creación de la siguiente generación. Los procesos de cruzamiento, mutación y evaluación de estos nuevos individuos son realizados por los algoritmos esclavos.

Realizado el proceso de selección, el proceso Maestro envía las parejas de los individuos seleccionados incluidas en el *matepool* a cada uno de los procesos esclavos. Estos procesos esclavos recombinan, mutan y evaluan estas parejas de cromosomas y regresan al proceso maestro la información de los nuevos individuos; es decir, nuevos cromosomas, con sus correspondientes valores de adaptabilidad y su conjunto de parámetros.

Este modelo se implementó en su topología de *árbol*. La topología de árbol empleada difiere de la presentada en la figura 6.2. en el punto de ser ésta una topología de árbol binario, como se muestra en la figura 6.4. El proceso maestro o proceso controlador está en el procesador raíz del árbol.

El pseudocódigo en OCCAM de esta implementación se anexan en el apéndice D. Además del código del programa se tiene el código de configuración (mapeo de procesos a

procesadores) de la red de transputers en ambos casos.

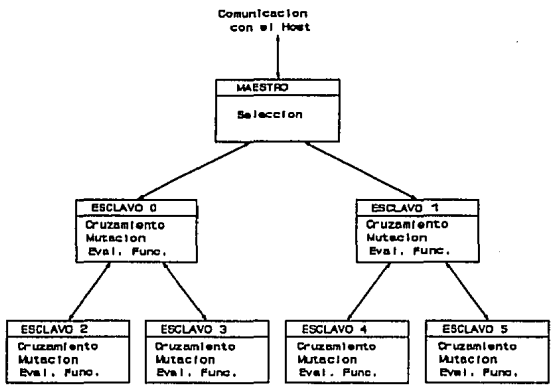


Figura 6.4. Arbol Binario.

VI.5.2. Algoritmo Genético Paralelo Modelo de Migración.

La teoría previa del algoritmo de migración nos lleva directamente a una implementación que explota su paralelismo geométrico. En este modelo, cada subpoblación es un proceso separado. La topología elegida para este modelo es un ***anillo de subpoblaciones***, teniendo la migración de individuos en una sola dirección alrededor del anillo (Figura 6.5).

El método de migración puede aplicarse al enviar de cada subpoblación un número de individuos, seleccionados *aleatoriamente*. El número de individuos recibidos debe ser igual al número de individuos enviados. En este algoritmo se introducen dos nuevos parámetros: ***migrate***, que corresponde al porcentaje de la población de individuos seleccionados para la migración de cada subpoblación en el tiempo de migración, y ***miginterval***, que determina el número de generaciones entre cada migración. A estos dos parámetros podemos darles los valores de $migrate=1/N_{subpop}$ y $miginterval=1$, donde ***migrate***, para este valor, nos marca que por cada subpoblación, en el tiempo de migración, un individuo migrará de una subpoblación a otra; y ***miginterval*** especifica que la migración se va a producir cada generación.

El emigrante de cada subpoblación es elegido bajo un criterio previamente establecido, el cual va a substituir a un individuo de la subpoblación a la cual va a migrar.

En la figura 6.5. se presenta la estructura del proceso del AGP de Migración con cada una de las subpoblaciones (AG Nodal) asignadas a un solo procesador. Se tiene además un proceso de interfase con el Host para los procesos de E/S a la máquina host. El proceso de interfase con el host está conectado a uno de los nodos, *Algoritmo Genético Monitor*, el cual pasa los resultados enviados de las subpoblaciones del anillo.

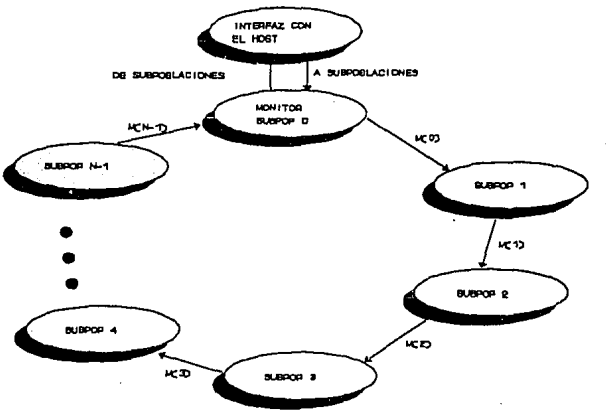


Figura 6.5. GA Migración, N subpoblaciones, Topología Anillo.

VI.5.3. Algoritmo Genético Paralelo Modelo de Difusión.

Al igual que en el modelo de migración, también difusión nos lleva directamente a una implementación que explota el paralelismo geométrico del algoritmo.

Para la implementación de este modelo podemos tomar la topología presentada en la figura 6.3. (toroidal). Otra opción es tomar el arreglo de procesadores en forma de *anillo*, con lo cual tendríamos la misma topología empleada para el modelo de migración. La diferencia entre estos dos modelos será que para el caso de difusión, la comunicación del anillo va a ser en ambos sentidos y no se va a tener una migración de individuos de una subpoblación a otra. En cada uno de los AGN del anillo se podrá tener acceso a la selección de otros individuos de la

población total pertenecientes a sus nodos vecinos además de los individuos residentes en el nodo, con lo cual se va a presentar un traslape en las subpoblaciones. Así, el número de posibles individuos para contribuir en la generación de nuevos miembros dentro de una subpoblación, será mayor al número de nuevos miembros que ésta debe generar.

En este **AGP Modelo de Difusión** las subpoblaciones, a su vez, estarán divididas en partes, las cuales serán enviadas a las subpoblaciones vecinas para la creación de los descendientes. Cada una de estas partes está constituída por más de un elemento. Con lo anterior, estamos reduciendo el intervalo de comunicación en cada proceso. Esto maximiza el uso eficiente del procesamiento paralelo. En la figura 6.6. se presenta gráficamente este envío y recepción de información entre subpoblaciones vecinas.

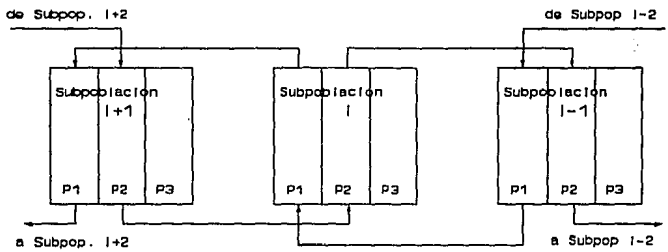


Figura 6.6. Intercambio de Información entre Subpoblaciones Vecinas.

El arreglo de los procesadores en una topología anillo con una comunicación bidireccional se presenta en la figura 6.7. El recuadro en esta figura nos marca la ubicación de todos los posibles padres de los nuevos miembros de la subpoblación *I*, lo cual 5/3 veces mayor al tamaño de la subpoblación. Esto es aplicable a cada uno de los nodos del sistema.

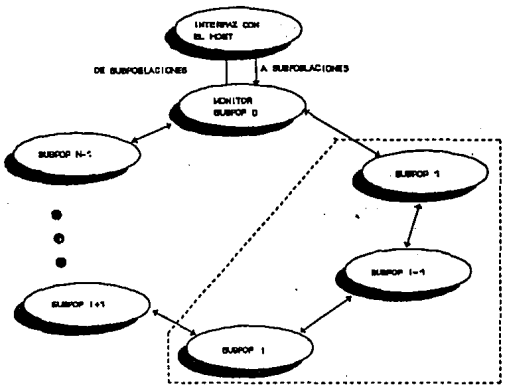


Figura 6.7. Anillo Bidireccional para el Modelo de Difusión

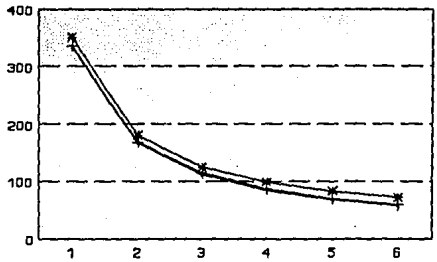
VI.6. RESULTADOS Y ANALISIS.

Para verificar la eficiencia, validez y escalabilidad de los tres modelos de Algoritmos Genéticos Paralelos se tomó como base el problema de Estimación Espectral de la señal Doppler simulada que fue analizada en el capítulo V. Para tal efecto se tomaron las mismas probabilidades de cruzamiento (P_c=*1.0*) y de mutación (P_m=*0.01*), así como el orden del modelo del estimador siendo éste igual a p=*6* y un tamaño de la población de *popsize=60*.

Las métricas más comunes para medir el desempeño de un sistema paralelo son el *tiempo de ejecución* y la *eficiencia*, las cuales fueron utilizadas para dicho propósito en este capítulo.

La implementación de los modelos paralelos ha sido realizada hasta 6 procesadores. En las figuras 6.8, 6.9 y6.10 se presentan el tiempo de ejecución, la eficiencia y la fracción serial, respectivamente, de las modelos de paralelismo implementados.

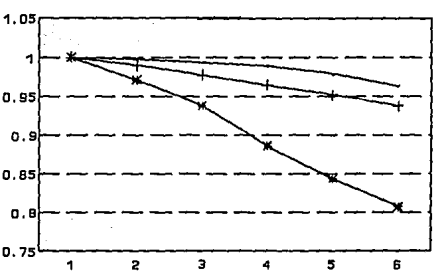
Tiempo de Ejecución [ms]



MIGRACION	—	333.624	167.104	111.936	84.356	68.159	57.752
DIFUSION	+	335.04	169.216	114.28	86.86	70.412	59.568
FARMING	*	350.52	180.532	124.584	98.92	83.135	72.384

Figura 6.8. Tiempo de Ejecución.

Eficiencia



MIGRACION	—	1	0.9982	0.9934	0.9887	0.9789	0.9628
DIFUSION	+	1	0.9899	0.9772	0.9643	0.9517	0.9375
FARMING	*	1	0.9708	0.9379	0.8859	0.8433	0.8071

Figura 6.9. Eficiencia.

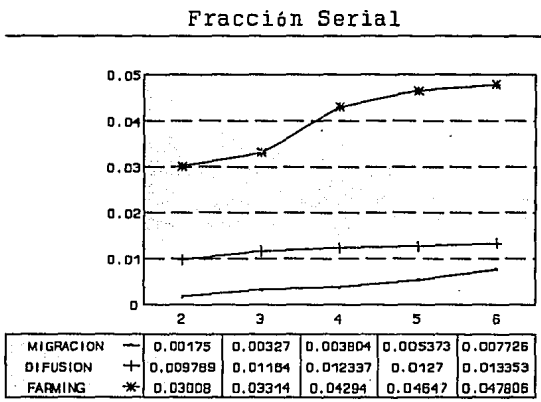


Figura 6.10. Fracción Serial.

En la figura 6.8 puede observarse que, el tiempo de ejecución en los tres modelos decrece asintóticamente, presentando migración y difusión tiempos muy semejantes, y mucho mayores que el modelo farming. Esta diferencia en el tiempo de ejecución está en relación a la longitud de la trayectoria de comunicación y los tiempos de envío y recepción de información del proceso maestro a los procesos esclavos que presenta el modelo farming.

Para los modelos de difusión y migración la topología empleada es igual, con la diferencia que, el modelo de difusión presenta una comunicación bidireccional además de manejar un mayor flujo de información que el modelo de migración. Esto hace que los tiempos presentados por el modelo de difusión sean un poco mayores que en el modelo de migración.

La figura 6.9 presenta la eficiencia de los sistemas paralelos implementados. Como se mencionó en el capítulo II, la eficiencia en un sistema paralelo es una métrica que nos indica el aprovechamiento que se tiene del hardware disponible. Para el cálculo de la eficiencia se parte de la ecuación 6.1, en la cual T_1 es el tiempo que toma un solo procesador en ejecutar el Algoritmo Genético para cada uno de los modelos, N es el número de procesadores, y T_N es el tiempo que toman N procesadores en ejecutar el Algoritmo Genético Paralelo, para cada uno de los modelos propuestos.

Los resultados muestran que con el aumento en el número de procesadores, la eficiencia del sistema disminuye por el incremento en las comunicaciones. Para el modelo farming, la eficiencia cae más rápidamente en comparación con los modelos de difusión y migración. Uno de los factores que afectan el comportamiento de la eficiencia es la relación entre el tiempo de procesamiento y el tiempo de comunicaciones correspondiente al envío y recepción de información. El hecho de que el proceso de selección sea realizado exclusivamente en el procesador maestro incrementa el número de comunicaciones con los esclavos, manteniendo en ciertas etapas del proceso, algunos procesadores esclavos inactivos.

La eficiencia de los modelos paralelos refleja además el grado de escalabilidad del sistema implementado. Los modelos de difusión y migración presentan una eficiencia por encima del 93% y del 96%, respectivamente, para 6 procesadores. Estas cifras altas muestran que estos modelos son eficientemente escalables teniendo tantas subpoblaciones como procesadores configuren el sistema.

Analizando los resultados presentados en la figura 6.10 que corresponden a la fracción serial, éstos nos indican dos aspectos importantes del sistema paralelo en consideración: el balance de cargas (distribución de tareas) y las comunicaciones entre procesadores. De esta manera, tenemos que, idealmente la fracción serial debe mantenerse constante con el incremento en el número de procesadores del sistema. En un caso real, el utilizar un mayor número de procesadores implica un incremento en el tiempo de comunicaciones, reflejándose en un incremento en la fracción serial al aumentar el número de procesadores.

En la figura 6.10, se observa que la fracción serial para el modelo de migración es menor a la presentada por difusión y farming. Estos niveles en la fracción serial están en función de las comunicaciones presentadas por cada modelo. Como se mencionó anteriormente, el flujo de información para el modelo de difusión es mayor que en el modelo de migración además de tener una comunicación bidireccional.

En la figura 6.10 también se observa un incremento pronunciado en la fracción serial cuando pasamos de 3 a 4 procesadores para el caso del modelo farming, ésto significa un desbalanceo del sistema implementado. En este modelo, la distribución de las tareas está relacionada con la distribución de las parejas de individuos que se van a recombinar en los N procesadores esclavos que se empleen. Considerando una población de 60 elementos, tenemos cada generación 30 pares de individuos que serán recombinados. Estas parejas de individuos pueden ser distribuidas adecuadamente cuando se cuenta con 2, 3, 5 y 6 procesadores. En el caso de tener 4 procesadores, tendríamos un procesador esclavo con un par de individuos más para la recombinación. Aunque al inicio del proceso ésto es irrelevante, al final todos los procesadores terminarán sus tareas, con excepción del procesador con el par de elementos extra, lo cual ocasiona un desbalance y consecuentemente una reducción en la eficiencia en nuestro sistema paralelo.

VI.7. CONCLUSIONES.

En la actualidad, la funcionalidad de los Algoritmos Genéticos ha sido extendida con versiones paralelas de los mismos. Esto con la finalidad de tener una mejor aproximación o un mayor acercamiento a lo que es su modelo natural.

En este capítulo se analizaron tres modelos de paralelismo de Algoritmos Genéticos: *farming, migración y difusión.*

Los resultados obtenidos revelan una reducción del tiempo de ejecución al utilizar los modelos paralelos de Algoritmos Genéticos, comparados con la versión secuencial del AG. El modelo de migración presenta la mayor eficiencia estando, en el caso del uso de 6 procesadores, por arriba del 96%, siguiendo el modelo de difusión con una eficiencia del 93%, y finalmente farming presentando una eficiencia del 80%.

Otras métricas como la fracción serial han sido utilizadas para el análisis de los modelos paralelos implementados, revelando desbalanceos en el modelo farming y por tanto una reducción considerable en su posibilidad de ser escalado. No obstante, el uso de técnicas de asignación dinámica de actividades (considerando estudios cuidadosos de granularidad), mejorarían considerablemente su respuesta.

Por otra parte, los modelos de difusión y migración no solamente presentan una alta escalabilidad, sino que al evolucionar subpoblaciones en forma independiente en cada procesador, la calidad de la solución obtenida mejora notablemente. En este sentido, el modelo de difusión presenta una mejor respuesta al modelo de migración, ya que al considerar una población local mayor (incluyendo elementos de procesadores vecinos), contiene una mayor diversidad en dicha población.

VI.8. REFERENCIAS.

[1] East, I.R., Macfarlane, D., "Implementation in OCCAM of Parallel Genetic Algorithms on Transputer Network", Parallel Genetic Algorithms: Theory and Applications, Joachim Stender (ed.), IOS Press, 1993.

[2] Flemming, P.J., "Parallel Algorithms and Genetic Algorithms", Taller Internacional de Procesamiento Paralelo, Departamento de Electrónica y Automatización, Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, Universidad Nacional Autónoma de México, 1994.

[3] Flemming, P.J., Chipperfield, A., Pohlheim, H., Fonseca, C., "Genetic Algorithms. Toolbox.

User's Guide.", University of Sheffield, 1994.

[4] Galletly, J., "OCCAM 2", Pitman Publishing, 1990.

[5] Gorges-Schleuter, M., "Comparison of Local Mating Strategies in Massively Parallel Genetic Algorithms", Parallel Problem Solving from Nature, Männer, R., Manderick (Ed.), 1992.

[6] Hoare, C.A.R., "Communicating Sequential Processes", Prentice Hall, 1985.

[7] Holland, J.J., "Adaptation in Natural and Artificial Systems", Ann Arbor, MI: University of Michigan Press, 1975.

[8] Inmos, Limited, "Transputer Overview. The Transputer Databook.", 2ª Edición, 1989.

[9] Karp, A.H., Flatt, H.P., "Measuring Parallel Processor Performance", Vol. 3, Num. 5, Pag. 539-543, 1990.

[10] Pettey, C.B., Leuze, M.R., "A Theoretical Investigation of Parallel Genetic Algorithms" Proceedings of Third International Conference on Genetic Algorithms, San Mateo, Ca: Morgan Kaufmann Publishers, Pag. 398-405, 1989.

[11] Schulze-Kremer, S., "Genetic Algorithms for Protein Tertiary Structure Prediction", Parallel Genetic Algorithms: Theory and Applications, Joachim Stender (ed.), IOS Press, 1993.

[12] Shumeet, B., "Structure and Performance of Fine-Grain Parallelism in Genetic Search", Proceedings of Fifth International Conference on Genetic Algorithms, 1993.

[13] Tanese, R., "Distributed Genetic Algorithms", Proceedings of Third International Conference on Genetic Algorithms, San Mateo, Ca: Morgan Kaufmann Publishers, Pag. 434-439, 1989.

CAPITULO VII

CONCLUSIONES

VII. 1. CONCLUSIONES GENERALES.

El objetivo general del trabajo presentado en esta tesis ha sido cumplido satisfactoriamente, al poder utilizar eficientemente el paralelismo inherente a los ***Algoritmos Genéticos***, tanto para disminuir el tiempo de ejecución como para mejorar la calidad de la solución encontrada. La alternativa propuesta planteó el objetivo de reducir la complejidad de cálculo computacional que presenta el algoritmo de covariancia modificada, utilizando para ésto la simplicidad y robustez asociada a la solución genética.

Los Algoritmos Genéticos, métodos de búsqueda y optimización basados en los mecanismos de la evolución y la genética natural, parten de una población inicial generada aleatoriamente que representa un conjunto de posibles soluciones al problema considerado. La reproducción de los individuos más aptos de una población, combinado con operadores genéticos tomados del medio natural, crean un mecanismo robusto que permite evolucionar estos individuos adaptándolos al problema hasta obtener la solución óptima. No obstante, el uso de poblaciones finitas y puntos de muestreo, inevitablemente generan errores que pueden conducir a valores lejanos al óptimo.

Por tal motivo, un estudio de los efectos de diferentes parámetros y esquemas de reproducción en el desempeño de los Algoritmos Genéticos fue llevado a cabo. Basados en una función exponencial creciente se analizaron tres métodos de selección: Ruleta, Escalamiento Lineal y Ranking. Asimismo, los efectos de las diferentes probabilidades de los operadores de mutación y cruzamiento, así como variaciones en el tamaño de la población fueron analizadas. Este estudio reveló que la rapidez de convergencia a la solución óptima de un método de selección en particular, depende de que tan eficientemente dicho método pueda mantener la diversidad entre los miembros de la población y así evitar la pérdida de elementos que pudieran ser fundamentales en alcanzar el valor óptimo. El método Ranking presenta estas características convergiendo al óptimo en pocas generaciones. Los resultados obtenidos también demostraron que la probabilidad de mutación en un Algoritmo Genético debe ser muy pequeña estando por debajo del 0.1%. En la probabilidad de cruzamiento se observó que ésta puede estar arriba del 75%, estando en función del método de selección empleado.

Realizado este estudio y con la finalidad de reducir la complejidad de implementación que presenta el método espectral paramétrico, se realizó de manera directa la implementación del estimador mediante Algoritmos Genéticos, substituyendo la función exponencial utilizada en el estudio descrito, por la función de error predictivo definida por el modelo del estimador. A fin de determinar el desempeño de la solución genética propuesta, se analizó una señal conocida de la cual se presentó su espectro teórico. En este análisis se varió el modelo del estimador, observando una respuesta en frecuencia más cercana a la teórica con un modelo de orden 10 en relación a un modelo de orden 6. Los resultados mostraron soluciones similares para el caso genético y de covariancia modificadas, superando ambos al método de FFT.

La solución genética demostró además que con pocas generaciones podemos tener una buena estimación espectral de la señal, debido al *paralelismo implícito* que presentan los AG's al explorar diferentes regiones del espacio de soluciones simultáneamente reduciendo el costo computacional.

Con los resultados obtenidos, este método espectral fue aplicado al análisis de señales Doppler, realizando un estudio comparativo de la respuesta en frecuencia presentada por el método no paramétrico de la Transformada Rápida de Fourier, el método paramétrico de covariancia modificada y la solución genética. Los resultados revelaron una respuesta espectral semejante en relación a valores de magnitud, frecuencia central y ancho de banda. En el caso de la solución genética, esta respuesta se obtuvo en tan solo tres generaciones.

Con el fin de reducir el tiempo de ejecución del algoritmo genético y explotar eficientemente la naturaleza paralela del mismo, se desarrollaron diversos modelos. Herramientas de hardware y software para el desarrollo de sistemas paralelos como son el *transputer* y el lenguaje de programación de alto nivel *OCCAM* fueron utilizados para la implementación de los mismos. Tres modelos de ***Algoritmos Genéticos Paralelos*** fueron implementados: *Migración, Difusión* y *Farming*. Los resultados obtenidos mostraron una eficiencia del 96% para el modelo de migración, 93% para el modelo de difusión y una eficiencia del 80% para el modelo farming utilizando para ello seis procesadores. Estas cifras muestran claramente el alto grado de escalabilidad de los modelos de difusión y migración, no siendo el caso para el modelo farming.

El cálculo de la fracción serial fue una importante herramienta de análisis de los sistemas paralelos implementados, revelando importante información en relación a la distribución de cargas y al proceso de comunicaciones. Estos resultados mostraron un desbalanceo de cargas en el modelo farming para el caso de 4 procesadores explicando la baja eficiencia de este modelo. La fracción serial presentó los diferentes niveles de comunicación de cada uno de los modelos paralelos, así como el aumento en los intervalos de comunicaciones cuando el número de procesadores del sistema paralelo fue incrementado, presentando el modelo de migración los niveles más bajos de esta métrica.

Por otra parte, cabe mencionar que debido a las características de la función de error predictivo utilizada en el método paramétrico, ésta posee una complejidad asociada al orden del modelo empleado para la estimación espectral. Se ha comprobado que para esta función

en particular, diferentes conjuntos de parámetros pueden producir errores muy cercanos a cero, lo cual nos indica la presencia en el espacio de soluciones de varios mínimos locales.

Para el caso en que se utiliza la solución de la matriz de covariancia, ésta encuentra una solución única dada por la formulación de sus ecuaciones y por tanto el error, aún cuando muy cercano a cero, no podrá mejorarse. Para el caso genético, el algoritmo explora diferentes regiones simultáneamente encontrando mejores respuestas.

Más aún, la existencia de soluciones semejantes en diferentes regiones, es explotada eficientemente mediante el uso de los Algoritmos Genéticos Paralelos, evolucionando cada procesador un Algoritmo Genético Nodal y, para el caso de los modelos de difusión y migración, intercambiando información importante sobre la solución.

Este trabajo ha presentado la aplicación de los Algoritmos Genéticos al problema de análisis espectral en Flujometría Doppler. No obstante, estos algoritmmos no son exclusivos de esta aplicación, con lo cual se presenta un método alternativo para la solución a problemas más generales.

VII.2. TRABAJO FUTURO.

El contenido de este trabajo de tesis plantea varios puntos para un trabajo futuro.

El método alternativo presentado ha sido planteado en base a un Algoritmo Genético Simple, el cual involucra un esquema de reproducción y los operadores genéticos de cruzamiento (empleando el cruzamiento en un solo punto) y mutación. El empleo de diferentes formas de implementación de estos operadores, el estudio de otros esquemas de reproducción y sus efectos en el comportamiento del AG, así como el estudio de otros operadores genéticos constituyen un punto importante de estudio.

En el campo de los Algoritmos Genéticos Paralelos se tomaron tres modelos para su implementación. Estas versiones de AGP's plantean el estudio de otros modelos paralelos que nos permitan aún más reducir el tiempo de procesamiento, además de explorar diferentes topologías y métodos de asignación de actividades.

Estos algoritmos han sido desarrollados sobre una arquitectura homogénea (sistema basado en transputers). Estos mismos algoritmos pueden ser mapeados para ejecutarse en una arquitectura heterogénea de transputer-DSP, analizando su eficiencia y desempeño para realizar un estudio comparativo con la arquitectura homogénea utilizada.

La disponibilidad de sistemas paralelos y la alternativa genética presentada abre la posibilidad de implementación de métodos de estimación espectral más complejos que nos permitan mejorar aun más la resolución en frecuencia del espectro para el caso de estudio.

APENDICE A

RUTINAS EN MATLAB PARA LA IMPLEMENTACION DE UN ALGORITMO GENETICO

% Función flipm(x): flipm.m
% De acuerdo al parámetro 'x' (probabilidad), retorna '0' o '1'.

```
function y=flipm(x)
s=rand;         % Genera un número aleatorio entre 0 y 1.
if s<x
        y=1;
else
        y=0;
end
```

% Función decode(a, b): decode.m
% Decodifica una cadena de bits pasando como parámetros la cadena de bits 'a' y % la longitud de la misma 'b' y regresando un valor real.

```
function y=decode(a,b)
accum=0.0;
powerof2=1;
for k=b:-1:1
        if a(k)==1
                accum=accum+powerof2;
        end
        powerof2=powerof2*2;
end
y=accum;
```

```
% Función rnd(z): rnd.m
% Determina un valor aleatoriamente entre '1' y 'z'.

function y=rnd(z)
s=rand*z;
y=round(z)+1;
```

```
% Función objfun(x): objfun.m
% Función objetivo

function y=objfun(x)
n=10;
coef=1073741824;
y=exp(n*logm(x/coef));
```

```
% Función select(a, b, c): select.m
% Función que selecciona por el Método de la Ruleta los individuos a
% recombinarse, donde 'b' es la suma de los valores de adaptabilidad
% (sumfitness), 'a' es el tamaño de la población (popsize) y 'c' es un vector  % que presenta el
valor de adaptabilidad de los individuos.

function y=select(a, b, c)
partsum=0.0;          % Suma parcial de adaptabilidad inicializada en '0'.
l=0;                  % Variable de individuo seleccionado inicializada.
rando=rand*b;         % Número aleatorio por la suma de adaptabilidad.
while (partsum<rando) & (l<a)
      l=l+1;          % Mientras 'partsum' sea menor a 'rando' y
      partsum=partsum+c(l);    % 'l' menor a 'a'.
end
y=l;
```

```
% preescal.m
% Rutina para el cálculo de los coeficientes 'a' y 'b' para el método de
% selección de escalamiento.
```

```
fmult=2.0;              % Número de copias para el mejor miembro.
if (min>((fmult*avg)-max)/(fmult-1.0))
        delta=max-avg;
        a=((fmult-1.0)*avg)/delta;
        b=(avg*(max-(fmult*avg)))/delta;
else
        delta=avg-min;
        a=avg/delta;
        b=-min*avg/delta;
end
```

```
% Función y=escala(aa, bb, x): escala.m
% En base a los coeficientes 'a' y 'b' calculados en el proceso de 'preescal' se
% calcula en nuevo valor de adaptabilidad. Se pasan los parámetros 'aa', 'bb' y
% 'x', el cual es el valor de adaptabilidad presente.

function y=escala(aa, bb, x)
        y=x*aa+bb;
```

```
% escalpop.m
% Rutina para el escalamiento de la población y cálculo del nuevo valor de la
% suma de todos los valores de adaptabilidad 'sumfitness'.

preescal;                       % Corre la rutina 'preescal' para determinar 'a' y 'b'.
sumfitness=0.0;
for l=1:popsize
   fitness1(l)=escala(fitness(l),aa,bb);  % Cálculo del nuevo valor de fitness.
   sumfitness=sumfitness+fitness1(l);          % Suma de los nuevos valores de
end                                                    % fitness.
```

```
% sorting.m
% Rutina de ordenamiento para el método de selección Ranking, ordenando la
% población en forma descendiente.


x=0;
```

```
while (x < popsize),
        x=x+1;
        for i=1:popsize-1,
        if (fitness(i)<fitness(i+1)),
                        cromn(i,:)=crom(i,:);
                crom(i,:)=crom(i+1,:);
                        crom(i+1,:)=cromn(i,:);
                        fitnessn(i)=fitness(i);
                        fitness(i)=fitness(i+1);
                        fitness(i+1)=fitnessn(i);
        end
        end
end
```

% ranking.m

```
% Rutina para el método Ranking la cual determina el vector con los índices de
% los individuos que van a cruzarse de acuerdo a la posición que guardan en la
% población ordenada.

maxc=popsize*0.2;              % Máximo número de copias para el mejor individuo
counter=0;
i=1;
while i<popsize
        if fitness[i]<=avg     % Individuo con fitness igual al promedio
                medio=i;         % le asigna una copia
                i=popsize+1;
        else
                i=i+1;
        end
end
ar[medio]=1;
counter=counter+1;             % Incrementa contador de copias
const=(1-maxc)/(medio-1);      % Calcula constante en base al promedio y al máximo
i=1;                                   % número de copias para el mejor elemento
while i<medio
        ar[i]=trunc(((const*(i+1))-const)+maxc);  % A cada uno de los individuos
        counter=counter+1;                        % arriba del promedio, les
        if counter>popsize                        % asigna tantas copias como
                dec=counter-popsize;              % se indique decreciendo de
                ar[i]:=ar[i]-dec;                       % acuerdo a la lista ordenada
```

```
            i:=medio+1;                         % Si se excede el tamaño de la popsize,
        else                                    % elimina el sobrante.
            i=i+1;
        end
end
for i=(medio+1):popsize                         % Si con la asignación de las copias a los
        if counter>popsize              %   individuos arriba del promedio 'counter'
            ar[i]=0;                            % es menor a 'popsize', otorga una copia a
        else                                    % cada uno de los individuos abajo del
            ar[i]=1;                            % promedio hasta que 'counter'='popsize'.
            counter=counter+1;
        end
end
i=1;
for k=1:popsize                                 % Crea un vector 'eleccion' con los índices
        while (ar[k]>0 & i<=popsize)            % de los indiviuos a los que se les asignaron
            eleccion[i]=k;              % copias.
            ar[k]=ar[k]-1;
            i=i+1;
        end
end
```

% función seleccio(a,b): seleccio.m

```
% Función de selección para el método ranking, teniendo como parámetros 'a' que
% corresponde al tamaño de la población el cual se va decrementando en cada
% llamada a la función, y 'b' que corresponde al vector 'eleccion' calculado en
% la rutina ranking.m que tiene los índices de los individuos que van a
% contribuir en la generación de la nueva población.

function [x,y,z]=seleccio(a,b) % Regresa 'x': popsize decrementada.
    elige=rnd(a);              %         'y': individuo elegido.
    y=b(elige);                %         'z': vector 'eleccion' eliminando los
    b(elige)=b(a);             %              valores ya seleccionados.
    b(a)=0;
    x=a-1;
    z=b;
```

% Función mutacion(a, pmutacion): mutacion.m

```
% Función que muta (cambia de estado) uno de los genes de un cromosoma. 'a'
% representa el gen a ser mutado y 'pmutacion' la probabilidad de mutación.

function y=mutacion(a, pmutacion)
mutante=flipm(pmutacion);  % Determina si será o no mutado.
if mutante
        y=~a;                          % Altera su estado de '0' a '1' o viceversa.
else
        y=a;
end
```

```
% entrada.m
% Procedimiento para la entrada de datos.

clc
popsize=input('Tamaño de la Población: ');
lcrom=input('Longitud del Cromosoma: ');
maxgen=input('Número de Generaciones: ');
pcross=input('Probabilidad de Cruzamiento: ');
pmutacion=input('Probabilidad de Mutación: ');
```

```
% inicialp.m
% Procedimiento que genera la población inicial.

crom=[];
fitness=[];
for i=1:popsize
        for j=1:lcrom
                crom(i,j)=flipm(0.5);          % Genera '1' o '0'.
        end
        x=decode(crom(i,:), l crom);           % Decodifica la cadena.
        fitness(i)=objfun(x);                  % Evalua la función objetivo.
end
```

```
% estadist.m
% Procedimiento para los cálculos estadísticos.
```

```
% Encuentra el valor de adaptabilidad mínimo, máximo y promedio, así como la
% suma de todos valores de fitness.

sumfitness=fitness(1);
min=fitness(1);
max=fitness(1);
for j=2:popsize
        sumfitness=sumfitness+fitness(j);
        if fitness(j)>max
                max=fitness(j);
        end
        if fitness(j)<min
                min=fitness(j);
        end
end
avg=sumfitness/popsize;
```

```
% crossove.m
% Procedimiento de recombinación (cruzamineto).
% Con los individuos padres ('mate1' y 'mate2') determinados por el esquema de
% selección), se elige un punto aleatorio entre '1' y 'lcrom-1' para el
% intercambio de material genético.

if flipm(pcross)
        jcross(lcrom-1);
else
        jcross=lcrom;
end
k=1:1:jcross;
        cromn(j,k)=mutacion(crom(mate1,k), pmutacion);
k=1:1:jcross;
        cromn(j+1,k)=mutacion(crom(mate2,k), pmutacion);
if jcross~=lcrom
        k=jcross:1:lcrom;
                cromn(j,k)=mutacion(crom(mate2,k), pmutacion);
        k=jcross:1:lcrom;
                cromn(j+1,k)=mutacion(crom(mate1,k), pmutacion);
end
```

```
% generaci.m
% Generación de descendientes de acuerdo al cruzamiento de dos individuos
% padres determinados por el esquema de selección y mutación de los mismos.

j=0;
while (j < popsize)
        j=j+1;
        if opc==3
                [total, mate1, eleccion]=seleccio(total, eleccion);
                [total, mate2, eleccion]=seleccio(total, eleccion);
        else
                mate1=select(popsize, sumfitness, fitness(:));
                mate2=select(popsize, sumfitness, fitness(:));
        end
        crossove                                % Llama proceso de cruzamiento
        x=decode(cromn(j,:), lcrom);
        fitnessn(j)=objfun(x);
        j=j+1;
        x=decode(cromn(j,:), lcrom);
        fitnessn(j)=objfun(x);
end
```

```
% principa.m
% Procedimiento Principal.

opc=menu('METODOS DE SELECCION','Ruleta','Escalamiento','Ranking');
gen=0;
entrada                 % Llama proceso 'entrada'.
inicialp                % Llama proceso 'inicialp'.
estadist                % Llama proceso 'estadist'.
while (gen < maxgen)
        gen=gen+1;
        if opc==2               % Si el esquema de selección elegido es
                preescal        % Escalamiento.
                escalpop
                fitness=fitness1;
        else                    % Si el esquema de selección elegido es
                sorting         % Ranking.
                ranking
```

```
        total=popsize;
    end                         % Si no, el esquema elegido es la Ruleta.
    generaci                % Llama proceso 'generaci'.
    estadist                % Llama proceso 'estadist'.
    vavg(gen)=avg;          % LLeva un registro de los valores de adaptabilidad
    vmax(gen)=max;      % mínimo, máximo y promedio de cada generación.
    vmin(gen)=min;
    crom(:,:)=cromn(:,:);               % Sustitución de población anterior
    fitness(:,:)=fitnessn(:,:);         % por nueva población.
end
```

APENDICE B

RUTINAS EN MATLAB PARA LA IMPLEMENTACION DEL METODO DE COVARIANCIA MODIFICADA MEDIANTE AG's

---

% entrada.m

```
% Rutina de entrada de datos.
% Además de las probabilidades de los operadores genéticos, el tamaño de la población y el
% máximo número de generaciones, también se proporciona el orden del modelo 'p', el
% número de datos 'n' y la secuencia de datos 'xx'
clc
popsize=input('Tamaño de la población: ');
lcrom=input('Longitud del cromosoma: ');
maxgen=input('Máximas generaciones: ')';
pcross=input('Probabilidad de crossover: ');
pmutacion=input('Probabilidad de mutacion: ');
fname=input('Archivo de datos: ','s');
eval(['load ',fname,'.dat']);
xx=eval(fname);
p=input('Orden del modelo: ');
n=input('Numero de datos: ');
```

% decodeca.m

```
% Como la aplicación es multiparámetros, el número de variables de decisión corresponde
% al número de subcadenas en las que debe dividirse el cromosoma de acuerdo al orden del
% modelo especificado.

columna=1;
renglon=1;
for jj=1:lcrom,
        if ((rem(jj,8)==1) & (jj~=1))        % Cada una de los parámetros es representado
                columna=1;                   % con 8 bits.
                renglon=renglon+1;            % Cada uno de los renglones de la matriz 'a'
        end                                  % representa cada uno de los parámetros.
```

```
        a(renglon,columna)=crom(j,jj);
        columna=columna+1;
end
for jj=1:p,                          % Los renglones de la matriz 'a' se decodifican
        aa(jj)=decodem(a(jj,:),8);   % para tener un valor entero no signado.
end
aa=(aa-128).* (5.0/256);             % A cada uno de los valores enteros no signados
parametro(j,:)=aa;                   % se les aplica una transformación para tener los
                                     % parámetros en un rango de [-2.5, 2.5].
```

% función objfunm(z, pp, nn, xx)

% Función objetivo que corresponde a la función de error predictivo del método de
% estimación espectral paramétrica de covariancia modificada, la cual se requiere minimizar.

```
function y=objfunm(z,pp,nn,xx)
    pf=0.0;                  % 'pf' corresponde al error predictivo en adelanto.
    pb=0.0;                  % 'pb' corresponde al error predictivo en atraso.
    for n=pp+1:nn,
        auxpf=xx(n);
        for kk=1:pp
            auxpf=auxpf+z(kk)*xx(n-kk);
        end
        auxpf=auxpf*conj(auxpf);
        pf=pf+auxpf;
    end
    pf=pf/(nn-pp);
    for n=1:nn-pp,
        auxpb=xx(n);
        for kk=1:pp
            auxpb=auxpb+(conj(z(kk))*xx(n+kk));
        end
        auxpb=auxpb*conj(auxpb);
        pb=pb+auxpb;
    end
    pb=pb/(nn-pp);
    pe=(pb+pf)/2;            % El error predictivo en adelanto y atraso es un promedio 'pf'
    y=1-(pe/.002);           % 'pb'. Como se requiere minimizar la función objetivo,
                             % se aplica un atransformación para obtener la función de
                             % adaptabilidad (Ec. 4.28), donde C_max=0.002.
```

APENDICE C

OCCAM 2 TOOLSET Y EL TRANSPUTER

C.1. INTRODUCCION.

Este apéndice es una recopilación de los aspectos más importantes para el manejo de la *Plataforma de Procesamiento Paralelo basada en Transputers.* Tiene como objetivo presentar una guía rápida de los pasos a seguir para la compilación, ligado, creación de archivos ejecutables y carga de un programa en *OCCAM* en un procesador de la plataforma, así como la programación para el uso de múltiples procesadores en paralelo.

Para mayor información referirse a [1][2].

C.2. OCCAM 2 TOOLSET.

OCCAM 2 Toolset es un sistema de desarrollo de software para la construcción y depuración de programas propios de los *transputers*. Este software permite crear programas para todos los tipos de transputers INMOS, además de una red de los mismos.

El desempeño del software desarrollado con estas herramientas se incrementa substancialmente mediante el uso de procesamiento paralelo, permitiéndonos construir sistemas paralelos de alto desempeño de una manera sencilla.

Modelo de Programación.

El modelo de promagración de OCCAM está basado en la comunicación de procesos paralelos a través de canales. Los canales conectan pares de procesos y permiten el intercambio de datos entre ellos. Cada uno de los procesos puede construirse a partir de un número de procesos paralelos, de tal manera que un sistema de software puede ser descrito como una jerarquía de procesos paralelos intercomunicados. Este modelo es consistente con muchos

La comunicación entre procesos es sincronizada. Cuando un mensaje es pasado entre dos procesos, el proceso de salida no se ejecuta hasta que el proceso de entrada está listo.

El modelo de programación de OCCAM también proporciona una fuerte base para la creación de sistemas utilizando diferentes lenguajes. Los compiladores ANSI C y FORTRAN son compatibles con OCCAM y pueden ser usados para la construcción de procesos OCCAM equivalentes en alguno de estos lenguajes.

En el lenguaje de promagración OCCAM el paralelismo puede ser expresado directamente. Cada proceso OCCAM es un proceso ejecutable independiente.

Confiabilidad.

El lenguaje OCCAM 2 puede ser extensivamente checado en tiempo de compilación, y muchos errores de programación pueden ser detectados antes de que el programa corra. Esto mejora significativamente la confiabilidad y hace la construcción de programas correctos más rápida y fácil.

Cada constructor en el lenguaje tiene un significado preciso. Esto hace a los programas fáciles de escribir y entender.

Programación en Tiempo Real.

OCCAM 2 proporciona una base sólida para la programación en tiempo real. Las características del transputer que permiten la programación en tiempo real son las siguientes:

* Implementación eficiente y directa de procesos paralelos en el hardware.
* Prioridad de proceso paralelos.
* Implementación de interrupciones de software como mensajes en canales de OCCAM, de tal manera que las rutinas de interrupción puedan ser escritas como procesos de alta prioridad.
* Fácil programación de timers.

oc - El compilador de OCCAM 2.

El compialdor de OCCAM 2 toma como entrada un código fuente contenido en un archivo texto de formato estandar. Cualquier editor de texto que produzca archivos ASCII puede ser usado para crear archivos fuentes de OCCAM.

El compilador soporta un número de directivas de códigos fuente los cuales permiten diferentes tipos de archivos fuentes para ser compilados conjuntamente. Las principales directivas son:

* **#INCLUDE**: incluye otros archivos fuente.
* **#USE**: usa separadamente códigos compilados y librerías.

Herramientas de Generación de Códigos.

Dos herramientas son usadas en secuencia para generar el archivo ejecutable para un transputer a partir del código objeto compilado:

ilink: Este ligador une módulos compilados separadamente y librerías en un solo archivo (unidad ligada).

icollect: Es usado para generar un código ejecutable para programas de un solo transputer a partir de unidades ligadas.

Carga de Código.

El código ejecutable es cargado en el transputer usando la herramienta del servidor de archivos del host **iserver**.

El servidor de archivos del host **iserver** es una herramienta combinada de cargador y servidor del host. Cuando se invoca para cargar un programa, se carga el código en el transputer y proporciona servicios durante el tiempo de corrida en el host (como un programa de I/O) para el programa del transputer. **iserver** corre en la computadora host, la cual no es usualmente un transputer.

ilibr.

Esta herramienta crea librerías de códigos compilados que pueden ser usadas en programas de aplicación. Cada archivo compilado separadamente, es dado como entrada a **ilibr**, dando como salida módulos de librerías que pueden ser ligados selectivamente.

Librerías de OCCAM.

La tabla 1. lista todas las librerías para el lenguaje OCCAM2.

LIBRERIA	DESCRIPCION
occamx.lib	Librerías del compilador.
hostio.lib	Librería de I/O de propósito general.
streamio.lib	Soporte de I/O.
snglmath.lib	Funciones matemáticas.
dblmath.lib	Funciones matemáticas de doble longitud.
string.lib	Rutinas de manipulación de cadenas.
convert.lib	Rutinas de conversión de tipos.

Tabla 1.

Constantes.

Los archivos que contienen las definiciones de constantes y protocolos se proporcionan para usarse con las librerías de OCCAM. Estas se listan en la tabla 2.

ARCHIVO	DESCRIPCION
hostio.inc	Ctes. del servidor de archivos del Host.
streamio.inc	Constantes de I/O.
mathvals.inc	Constantes matemáticas.
linkaddr.inc	Direcciones de los links del transputer.

Tabla 2.

El compilador automáticamente carga las librerías requeridas para un combinación específica de opciones del compilador.

Librerías Matemáticas.

Estas librerías proporcionan funciones trigonométricas y logarítmicas para todos los diferentes tipos de tansputers.

Librerías de I/O.

Hostio. Esta librería contiene rutinas que proporcionan el acceso al sistema de archivos y otros servicios del host vía el servidor de archivos. Las rutinas están basadas en la

comunicación con el servidor vía el protocolo **SP**. El protocolo **SP** está definido en el archivo **hostio.inc.**

La librería hostio es usada para:

* Manipulación de archivos.
* Acceso al host.
* Terminal I/O.

Streamio. Contiene rutinas las cuales proporcionan I/O en un nivel más alto que las rutinas hostio. El protocolo está basado en un modelo de flujo ('stream'). Esta librería es usada para I/O basada en caracteres usando protocolos de flujo ('stream'), y para controlar el despliegue de pantalla. Los protocolos para la librería streamio están definidos en el archivo **streamio.inc.**

convert.lib.

La librería de conversión de tipos convierte tipos de datos de OCCAM a cadenas ASCII y viceversa.

C.3. DESARROLLO DE PROGRAMAS.

Las principales etapas en el desarrollo de un programa para un transputer y las herramientas a usar en cada etapa se listan a continuación.

1. Escribir el código fuente. El código fuente puede ser escrito usando algún editor ASCII. El código puede dividirse en varios archivos fuentes.

2. Compilación. Cada archivo fuente debe ser compilado usando el compilador de OCCAM 2 oc para producir uno o más archivos objetos compilados. Cada archivo debe ser compilado para el mismo o un tipo de transputer o modo de compilación compatible. Esto es,

oc ***archivo*** **-t800**

donde ***archivo*** es el nombre del archivo que contiene el código fuente con extensión ***.occ***, y **-t800** es el tipo de transputer donde se correrá el programa. La salida generada por el compilador **oc** es un archivo con extensión **.tco**.

3. Ligar las unidades compiladas. Los archivos fuentes compilados son ligados para generar un solo archivo conocido como una unidad ligada. Esta operación también liga los módulos de las librerías requeridas por el programa, las cuales son seleccionadas por el tipo de

transputer y modo de compilación del código de librerías compiladas.

ilink ***archivo.tco*** **librería1.lib librería2.lib -t800 -f occam8.lnk**

donde ***archivo.tco*** es el archivo objeto compilado generado por **oc, librería1.lib, librería2.lib, etc.**, son las librerías requeridas por el programa, **-t800** es el tipo de transputer, la opción **-f** especifica un archivo indirecto ligador que contiene comandos y directivas para **ilink**. Se tienen tres archivos indirectos para diferentes tipos de transputers. Estos son **occam2.lnk, occama.lnk**

y **occam8.lnk**. Estos archivos identifican varias librerías incluyendo librerías del compilador las cuales son requeridas para ser ligadas con el programa.

El archivo **occam8.lnk** debe ser usado para los transputers serie **T800.**

La unidad ligada generada por **ilink** es un archivo con extensión **.lku.**

4. Creación del código ejecutable. Antes de que el programa pueda correrse se debe hacer 'bootable'. **icollect** es usado para crear el programa ejecutable usando la unidad ligada generada en el paso anterior como archivo de entrada.

icollect ***archivo.lku*** **-t**

donde ***archivo.lku*** es la unidad ligada, y la opción **-t** indica que el programa se va a correr en un solo transputer. El programa 'bootable' se va a escribir en el archivo ***archivo.btl.***

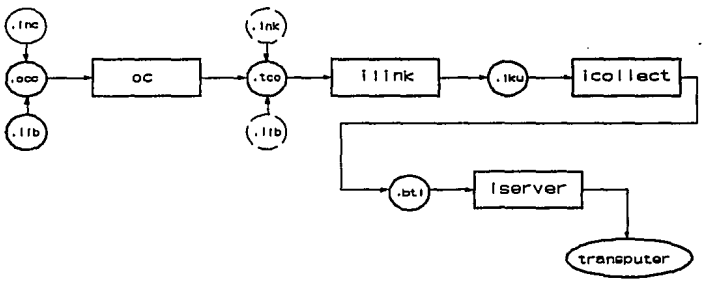
icollect también va a crear un archivo binario de configuración. Este archivo describe la configuración de la red que para nuestro caso será un solo transputer. Este archivo presenta la extensión **.cfb** y es creado para ser usado por el depurador.

5. Cargar y correr el programa. El archivo ejecutable es cargado en el transputer por medio del link del host. Una vez cargado en memoria, el código empieza a ejecutarse. Para cargarlo se usa el servidor de archivos del host **iserver.**

iserver -Puxdea2:# ***de site*** **-se -sb** ***archivo.btl***

donde **-Puxdea2** especifica la máquina hosts, ***:# de site*** especifica el site asignado para correr los programas. La opción **sb** especifica el archivo a ser ejecutado y carga el programa en el transputer. Tiene el efecto de inicializar el transputer ('reset'), abrir la comunicación con el host, y cargar el programa. La opción **se** lleva al servidor a terminar el programa si éste presenta errores. ***archivo.btl*** es el archivo ejecutable generado por **icollect.**

Resumiendo todos estos pasos de una manera gráfica tenemos la siguiente figura.



Arquitectura de Compilación de OCCAM 2 Toolset.

Creación de Librerías.

Para Crear una librería se requieren únicamente dos pasos:

1. Con el compilador de OCCAM 2 **oc**, compilar el archivo del código fuente de las librerías.

oc ***archivo***

archivo puede tener cualquier extensión.

2. **oc** genera como salida un archivo con la extensión **.tco**. Este archivo es la entrada para **ilibr**, dando como resultado un archivo con extensión **.lib**. Este archivo puede usarse dentro de la estructura de un programa en OCCAM mediante la directiva **#USE**.

ilibr ***archivo.tco***

C.4. CONFIGURACION DE REDES DE TRANSPUTERS.

Para la contrucción de programas para una red multitransputer un programa es dividido en un número de componentes, y cada una de éstos se implementa como un proceso en OCCAM. Cada proceso puede comunicarse con otros procesos residentes en el mismo transputer o, vía links, con procesos en otros transputers.

Estos programas deben ser configurados para correr en una red física de transputers. Para tal efecto se requiere de un archivo de configuración escrito en el lenguaje de configuración de OCCAM. Este archivo es creado por el usuario como un archivo texto.

Modelo de Configuración.

Este modelo consiste de las siguientes partes:

* Descripción de la red del hardware.
* Descripción del software.
* Mapeo entre los procesos y canales del software y los nodos (procesadores) y arcos (conexiones de los links de los transputers) de la red.

Lenguaje de Configuración.

Una descripción de configuración consiste de una secuencia de declaraciones e instrucciones. Estas son evaluadas por el compilador de OCCAM, el cual es llamado durante la configuración por la herramienta configurador **occonf**.

Las declaraciones de configuración introducen procesadores físicos y arcos de la red, conexiones de red y atributos del procesador, procesadores lógico a ser mapeados a procesadores físicos, la descripción del software, y el mapeo entre procesadores lógicos y físicos. A las descripciones de configuración generalmente le siguen las intrucciones las cuales desempeñan varias acciones relacionadas a las declaraciones. En las tablas 3 y 4 se presentan las descripciones de las declaraciones de configuraciones y los instrucciones de configuración, respectivamente.

DECLARACION	DESCRIPCION.
NODE	Declara procesadores (nodos de una gráfica). Pueden considerarse físicos si son definidos como parte de la descripción del HW, o lógicos si son definidos como parte de la descripción del SW y mapeados a procesadores físicos.
ARC	Declara las conexiones (arcos de una gráfica) entre procesadores (usando los links del transputer). Estas conexiones no necesitan ser declaradas como ARC's al menos que los canales requieran ser explícitamente colocados en unlink en particular.
NETWORK	Define las conexiones y atributos de los nodos declarados previamente (procesadores físicos).
MAPPING	Define el mapeo entre los procesadores lógicos y los procesadores físicos.
CONFIG	Declara la descripción del software.
EDGE	Define conexiones externas de la descripción del hardware. Los edges externos pueden ser el host o algún periférico conectado vía un adaptador.

Tabla 3.

INSTRUCCION	DESCRIPCION.
SET	Define valores para los atributos de NODE.
CONNECT	Define la conexión entre dos nodos.
MAP	Define el mapeo de un procesador lógico a un procesador físico declarado como un NODE.
PROCESSOR	Declara un proceso y lo asocia con un procesador lógico o físico.
DO	Agrupa una o más acciones definidas por los enunciados SET, CONNECT o MAP.

Tabla 4.

a) Descripción del Hardware.

Esta primera parte del modelo de configuración está consistituída por tres etapas.

♦ **Declaración de Procesadores:** Los procesadores son declarados para ser del tipo *NODE*, del mismo modo que la declaración de cualquier tipo de variable en OCCAM.

NODE worker:
[8]NODE pipeline:

♦ **Atributos del Nodo:** Un nodo tiene un conjunto de atributos. Se hace referencia a un atributo al relacionar el nombre del nodo con el nombre del atributo. Estos atributos son:

[]BYTE type: Tipo de transputer.
INT memsize: Tamaño de la memoria en byte's.
BOOL root: Define un procesador raíz si no hay conexión al Host.
INT romsize: Tamaño de la memoria ROM asignada al procesador.

♦ **Descripción de la Red:** La palabra reservada *NETWORK* introduce una sección, la cual describe la conectividad y atributos de los *NODE's* declarados previamente. Esto debe ser declarado fuera de la descripción *NETWORK*, de tal manera que sea visible dentro y fuera de la descripción *NETWORK*.

Para describir un solo procesador la palabra *SET* proporciona los valores para los atributos de ese procesador. Esto puede realizarse como una asignación múltiple o atributo por atributo. Como ejemplo tenemos el siguiente caso:

```
NETWORK simple
  SET procesador(type, memsize:="T800", 1024*1024)
  :
```

o,

```
NETWORK simple
  DO
    SET procesador(type:="T800")
    SET procesador(memesize:=1024*1024)
:
```

El constructor **DO** no implica algún orden en particular. Los atributos **type** y **memsize** deben ser definidos para todos los procesadores.

Si una red es configurada para ser cargada de ROM, el atributo **root** debe ser verdadero para un solo procesador. Por default es falso. El atributo **romsize** debe ser el número de byte's de ROM en el procesador raíz. Estos atributos son ignorados si la red es configurada para ser ejecutada desde el link.

Los procesadores deben ser conectados por *CONNECT* citando un par de links. Esto es:

```
VAL k IS 1024:
NETWORK dos
  DO
    SET proc1(type, memsize:="T800", 2048*k)
    SET proc1(root, romsize:=TRUE, 256*k)
    SET proc2(type, memsize:="T414", 1024*k)
    CONNECT proc1[link][0] TO proc2[link][3]
:
```

Conexión al Host: Se tiene un *EDGE* predefinido llamado **HOST**, el cual indica la conexión a una computadora host.

```
NODE simple:
ARC hostlink:
NETWORK comuhost
  DO
    SET simple(type, memsize:= "T800", 1000000)
    CONNECT simple[link][0] TO HOST WITH hostlink
:
```

Cuando se configura un programa el cual está diseñado para se ejecutado via un link de un transputer, un procesador debe estar conectado al *EDGE* predefinido (**HOST**).

b) Descripción del Software.

La descripción del software es un proceso OCCAM, **PAR** o **PLACED PAR**, con procesos descritos por la instrucción *PROCESSOR*. Esto identifica que procesos corresponden a cada uno de los procesadores.

Los **NODE's** pueden ser procesadores lógicos o físicos. Una instrucción *PROCESSOR* asocia el nombre de un proceso con el nombre correspondiente del procesador lógico o físico. Un mismo procesador puede ser referenciado en más de una instrucción *PROCESSOR*. El conjunto de procesos asignados a un procesador correrán concurrentemente en el mismo.

La facilidad de crear librerías de unidades ligadas (extensión **.lku**) proporciona un método fácil de asignar un proceso en diferentes tipos de procesadores en la descripción del software.

c) Descripción de Mapeo.

Una estructura *MAPPING* es usada si se han declarado procesadores lógicos. *MAPPING* mapea procesadores lógicos usados en la descripción del software en procesadores físicos usados en la descripción del hardware. Esta estructura de mapeo puede aparecer antes o después de la descripción del software.

Teniendo declarado los procesadores físicos, como parte de la descripción del hardware, y los procesadores lógicos, como parte de la descripción del software, podemos asignar los procesos lógicos a los físicos usando la instrucción *MAP*.

```
MAPPING map
  MAP proc.log ONTO proc.fis
:
```

Podemos también mapear una lista de procesadores lógicos a un procesador físico únicamente:

```
MAPPING map
  MAP proc.log1, proc.log2, proc.log3 ONTO proc.fis
:
```

Teniendo definidas las tres partes que constituyen el modelo de configuración para una red de transputers se presenta el ejemplo siguiente:

```
-- Descripción del Hardware.
VAL K IS 1024:
VAL M IS K*K:
NODE root.p, worker.p:
ARC hostlink:
NETWORK simple.red
  DO
    SET root.p(type, memsize:="T800", 8*M)
    SET worker.p(type, memsize:="T800", 8*M)
    CONNECT root.p[link][1] TO HOST WITH hostlink
    CONNECT root.p[link][2] TO worker.p[link][1]
:

-- Mapeo
NODE root.l, worker.l:
MAPPING
  DO
    MAP root.l ONTO root.p
    MAP worker.l ONTO worker.p
:

-- Descripción del Software.
#INCLUDE "proto.inc"      -- protocolo declarado
#USE "root.lku"           -- unidad ligada
#USE "worker.lku"                 -- unidad ligada
CONFIG
  CHAN OF SP fs:
  CHAN OF SP ts:
  CHAN OF protocol root.to.worker, worker.to.root:
  PLACED PAR
    PROCESSOR root.l
      root.process(worker.to.root, root.to.worker)
    PROCESSOR worker.l
      worker.process(root.to.worker, worker.to.root)
:
```

Este archivo de configuración debe tener la extensión **.pgm**. Para poder ejecutar este programa se necesita primeramente compilar con el compilador **oc** de OCCAM los dos archivos ***root.occ*** y ***worker.occ*** que corresponden a los procesos, ligarlos para generar los archivos con extensión **.lku** y poder ser incluidos en el archivo ***archivo.pgm***.

oc *root.occ*
oc *worker.occ*
ilink *root.tco* librerías.lib -t800 -f occam8.lnk
ilink *worker.tco* librerías.lib -t800 -f occam8.lnk

Teniendo los archivos ligados de los procesos, se procede a configurar el archivo **.pgm** con la herramienta **occonf**.

occonf ***archivo.pgm***

Este comando va a crear un archivo de salida con extensión **.cfb**. Para crear el código ejecutable se debe utilizar la herramienta **icollect**. Esto es:

icollect ***archivo.cfb***

lo cual va a generar un archivo ejecutable con extensión **.pgm**.

Para cargar y correr el programa se ejecuta la misma instrucción utilizada para correr un programa en un solo transputer descrita en la sección 3.

Este ejemplo se ilustra en la figura 2.

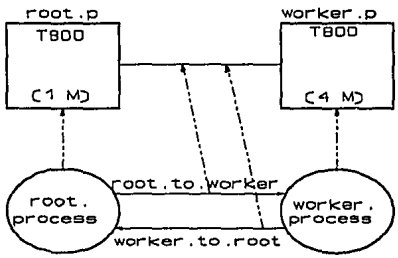


Figura 2. Programa de Ejemplo para dos Transputers.

Sumario.

Resumiendo, los pasos a seguir para construir un programa que corra en una red de transputers son los siguentes:

1. Especificar como será distribuído el programa en los transputers de la red.
2. Escribir la configuración para el programa (archivo **.pgm**):
   a) Descripción del hardware de la red.
   b) Descripción del software de la red.

1. Especificar como será distribuído el programa en los transputers de la red.
2. Escribir la configuración para el programa (archivo **.pgm**):
   a) Descripción del hardware de la red.
   b) Descripción del software de la red.
   c) Mapeo de procesadores lógicos a procesadores físicos.
3. Compilar todos los procedimientos que forman el código de cada uno de los transputers por separado.
4. Ligar cada uno de los procedimientos con sus componentes en un archivo con el nombre usado en la directiva **#USE** en el archivo fuente de configuración.
5. Correr el configurador **occonf** con el archivo de configuración con extensión **.pgm.**
6. Crear el código ejecutable con la herramienta **icollect.**
7. Cargar el programa en la red usando el servidor de archivos (**iserver**).

REFERENCIAS.

[1] OCCAM 2 Toolset. User Manual Part 1: User Guide ans Tools. Inmos.

[2] OCCAM 2 Toolset. User Manual Part 2: OCCAM Libraries and Appendices. Inmos.

APENDICE D

PSEUDOCODIGO DE ALGORITMOS GENETICOS PARALELOS Y ARCHIVOS DE CONFIGURACION PARA DIFERENTES MODELOS

Modelo de Migración.

El código que se presenta a continuación corresponde al Algoritmo Genético Nodal Monitor para el modelo de migración. El Algoritmo Nodal correspondiente a cada una de las subpoblaciones, es similar al que presenta el algoritmo monitor exceptuando la parte de datos iniciales. Estos datos son: tamaño de la población (*popsize*), número de generaciones (*maxgen*) y el segmento de datos correspondiente a la señal Doppler (*xx*).

```
-- ALGORITMO GENETICO NODAL. MODELO DE MIGRACION

... Include's
... Use's

PROC gam(CHAN OF SP fs, ts, CHAN OF INDIVIDUO from.ga, to.ga)

  ... Definición de Variables

 SEQ
   error:=TRUE
   ... Datos de Entrada
   to.ga ! popsize; maxgen; seed; n::xx
   ... Genera Población Inicial
   SEQ gen=0 FOR maxgen
     SEQ
       ... Seleccion
       ... Cruzamiento
       ... Mutacion
       ... Popn a Popv
       ... Estadistica
       PAR
         -- Proceso de migración de una subpoblación a otra
         to.ga ! crom[emigrante]; fitness[emigrante]
         from.ga ? crom[inmigrante]; fitness[inmigrante]
   -- Recibe mejor individuo de cada subpoblacion y elige el mejor
   to.ga ! max; parametro[vma]
   from.ga ? avg; parametros
```

El archivo de configuración correspondiente a las conexiones de los procesadores y el mapeo de procesos a procesadores para el modelo de migración se presenta a continuación:

```
-- ARCHIVO DE CONFIGURACION. MODELO DE MIGRACION.
-- HW
VAL K IS 1024:
VAL M IS K*K:
NODE        gam.p, ga5.p:                  -- Definición de procesadores físicos
[np-2]NODE  gaa.p:
ARC         hostlink:                      -- Definición de comunicación con el Host

NETWORK first
 DO
   SET gam.p(type, memsize:="T805",8*M)    -- Atributos de los Transputers
   SET ga5.p(type, memsize:="T805",8*M)    -- Tipo de transputes y tamaño de la memoria
   DO i=0 FOR np-2
     SET gaa.p[i](type, memsize:="T805",8*M)
   CONNECT gam.p[link][1] TO HOST WITH hostlink        -- Conexión con el Host.
   CONNECT gam.p[link][2] TO gaa.p[0][link][1]     -- Conexiones entre transputers.
   CONNECT gaa.p[0][link][3] TO gaa.p[1][link][0]
   CONNECT gaa.p[1][link][2] TO gaa.p[2][link][1]
   CONNECT gaa.p[2][link][3] TO gaa.p[3][link][0]
   CONNECT gaa.p[3][link][2] TO ga5.p[link][1]
   CONNECT ga5.p[link][3] TO gam.p[link][0]
:

-- mapping
NODE        gam.l, ga5.l:
[np-2]NODE  gaa.l:
MAPPING
 DO
   MAP gam.l ONTO gam.p                    -- Mapeo de procesos a procesadores
   DO i=0 FOR np-2                         -- (procesos lógicos a procesos físicos)
     MAP gaa.l[i] ONTO gaa.p[i]
   MAP ga5.l ONTO ga5.p
:

--SW
#INCLUDE    "hostio.inc"
#INCLUDE    "proto.inc"
#USE        "gam.lku"
#USE        "ga5.lku"
#USE        "gaa.lku"
CONFIG
  CHAN OF SP fs:                           -- Declaración de canales para la comunicación
  CHAN OF SP ts:                           -- con el Host
  PLACE fs, ts ON hostlink:
  [np]CHAN OF ANY mig:                     -- Canales para la comunicación entre procesad.
  PLACED PAR
    PROCESSOR gam.l
      gam(fs, ts, mig[np-1], mig[0])
    PAR i=0 FOR np-2
      PROCESSOR gaa.l[i]
          gaa(mig[i], mig[i+1])
```

```
  PROCESSOR ga5.f
    ga5(mig[np-2], mig[np-1])
:
```

Modelo de Difusión.

Para el modelo de difusión, al igual que para el modelo de migración, el Algoritmo Genético Nodal es similar al Algoritmo Genético Monitor sin tomar la parte de datos de entrada, con lo cual tenemos lo siguiente:

```
-- ALGORITMO GENETICO NODAL. MODELO DE DIFUSION.

... Include's
... Use's

PROC gam(CHAN OF SP fs, ts, CHAN OF ANY from.ga.up, from.ga.down, to.ga.up,
      to.ga.down)

 ... Definicion de Variables

 SEQ
   error:=TRUE
   neigh:=TRUE
   ... Datos de Entrada
   to.ga.down ! popsize; maxgen; seed; n::xx
   ... Genera Poblacion Inicial
   SEQ gen=0 FOR maxgen
     SEQ
       ... Seleccion
       IF
         neigh
           SEQ
             to.ga.down ! [crom[mate2] FROM 0 FOR lcrom]; fitness[mate2]
             from.ga.up ? [mate FROM 0 FOR lcrom]; fitt
             neigh:=FALSE
         TRUE
           SEQ
             to.ga.up ! [crom[mate2] FROM 0 FOR lcrom]; fitness[mate2]
             from.ga.down ? [mate FROM 0 FOR lcrom]; fitt
             neigh:=TRUE
       ... Cruzamiento
       ... Mutacion
       ... Popn a Popv
       ... Estadistica
   -- Recibe mejor individuo de cada subpoblacion y elige el mejor
   to.ga.down ! max; parametro[vma]
   from.ga.up ? avg; parametros
:
```

Comparando el modelo de difusión con el modelo de migración, podemos observar que éstos son muy semejantes, con la diferencia básica en comunicaciones entre un modelo y otro. En el modelo de difusión se tiene una comunicación bidireccional, especificándose los

canales como *from.ga.up* y *to.ga.up* para la comunicación con la subpoblación *I-1*, y los canales *from.ga.down* y *to.ga.down* para la comunicación con la subpoblación *I+1*. Además observamos que la fase de comunicación se presenta después del proceso de selección.

El archivo de configuración de la red y el mapeo de procesos a procesadores para este modelo, se presenta a continuación:

```
-- ARCHIVO DE CONFIGURACION. MODELO DE DIFUSION.
-- HW
VAL K IS 1024:
VAL M IS K*K:
NODE        gam.p, ga5.p:
[np-2]NODE  gaa.p:
ARC         hostlink:

NETWORK first
  DO
    SET gam.p(type, memsize:="T805",8*M)
    SET ga5.p(type, memsize:="T805",8*M)
    DO i=0 FOR np-2
      SET gaa.p[i](type, memsize:="T805",8*M)
    CONNECT gam.p[link][1] TO HOST WITH hostlink
    CONNECT gam.p[link][2] TO gaa.p[0][link][1]
    CONNECT gaa.p[0][link][3] TO gaa.p[1][link][0]
    CONNECT gaa.p[1][link][2] TO gaa.p[2][link][1]
    CONNECT gaa.p[2][link][3] TO gaa.p[3][link][0]
    CONNECT gaa.p[3][link][2] TO ga5.p[link][1]
    CONNECT ga5.p[link][3] TO gam.p[link][0]
:

-- mapping
NODE        gam.l, ga5.l:
[np-2]NODE  gaa.l:
MAPPING
  DO
    MAP gam.l ONTO gam.p
    DO i=0 FOR np-2
      MAP gaa.l[i] ONTO gaa.p[i]
    MAP ga5.l ONTO ga5.p
:

--SW
#INCLUDE    "hostio.inc"
#INCLUDE    "proto.inc"
#USE        "gam.lku"
#USE        "ga5.lku"
#USE        "gaa.lku"

CONFIG
  CHAN OF SP fs:
  CHAN OF SP ts:
  PLACE fs, ts ON hostlink:
  [np]CHAN OF ANY dif.d, dif.i:
```

```
 PLACED PAR
  PROCESSOR gam.l
    gam(fs, ts, dif.d[np-1], dif.i[np-1], dif.i[0], dif.d[0])
  PAR i=0 FOR np-2
    PROCESSOR gaa.l[i]
        gaa(dif.d[i], dif.i[(np-2)-i], dif.i[(np-1)+i], dif.d[i+1])
  PROCESSOR ga5.l
    ga5(dif.d[np-2], dif.i[0], dif.i[1], dif.d[np-1])
:
```

Modelo Farming.

A diferencia de los modelos de migración y difusión, farming presenta dos Algoritmos Genéticos diferentes: el AG maestro y el AG esclavo. La estructura que presenta el AG monitor es la siguiente:

```
-- ALGORITMO GENETICO MAESTRO O CONTROLADOR. MODELO FARMING.

... Include's
... Use's

PROC gac(CHAN OF SP fs, ts, CHAN OF ANY from.w1, from.w2, to.w1, to.w2)

  ... Declaracion de Variables

 SEQ
  error:=TRUE
  going:=TRUE
  ... Datos de Entrada
  ... Genera Poblacion Inicial
  SEQ gen=0 FOR maxgen
    SEQ
      ... Seleccion
      IF
        going
          PAR
            to.w1 ! [pool FROM 0 FOR (popsize/2)]; xx; seed; maxgen
            to.w2 ! [pool FROM (popsize/2) FOR (popsize/2)]; xx;
                  seed; maxgen
            going:=FALSE
        TRUE
          SEQ
            from.w1 ? [crom FROM 0 FOR (popsize/2)];
                    [fitness FROM 0 FOR (popsize/2)]
            from.w2 ? [crom FROM (popsize/2) FOR (popsize/2)];
                    [fitness FROM (popsize/2) FOR (popsize/2)]
      max, vma, avg, vmi:=ESTADISTICA(popsize, fitness)
:
```

En el AG controlador o maestro podemos ver que se realiza el procedimiento de selección creando un **'matepool'** con las parejas de los individuos seleccionados, y

posteriormente envía a los procesos esclavos parte de este **'matepool'** para que realicen el cruzamiento, mutación y evaluen los nuevos miembros, regresando estos datos al proceso controlador para realizar el proceso estadístico y pasar a la siguiente generación repitiendo este mismo proceso. En el AG esclavo se realiza, como se mencionó, el cruzamiento, mutación de las parejas recibidas del maestro, y la evaluación de los nuevos individuos. Para los procesos esclavos se presenta el siguiente pseudocódigo:

```
-- ALGORITMO GENETICO ESCLAVO. MODELO FARMING.
... Include's
... Use's
PROC gaw(CHAN OF ANY from.monitor, to.monitor, from.w1, from.w2, to.w1, to.w2)

  ... Declaracion de Variables

 SEQ
   going:=TRUE
   prime:=TRUE
   primr:=TRUE
   WHILE going
     SEQ
       IF
         primr
           PAR
             from.monitor ? [crom FROM 0 FOR popsize/2]; xx; seed; maxgen
             primr:=FALSE
         TRUE
           from.monitor ? [crom FROM 0 FOR popsize/2]; maxgen
       pop:=popsize/np
       IF
         prime
           PAR
             to.w1 ! [crom FROM 0 FOR pop]; xx; seed-2500(INT32); maxgen
             to.w2 ! [crom FROM pop FOR pop]; xx; seed-120(INT32); maxgen
             [pool FROM 0 FOR pop]:=[crom FROM (2*pop) FOR pop]
             prime:=FALSE
         TRUE
           PAR
             to.w1 ! [crom FROM 0 FOR pop]; maxgen
             to.w2 ! [crom FROM pop FOR pop]; maxgen
             [pool FROM 0 FOR pop]:=[crom FROM (2*pop) FOR pop]
       IF
         maxgen <= 0
           going:=FALSE
         TRUE
           SKIP
       ... Cruzamiento
       ... Mutación
         PAR
         from.w1 ? [cromn FROM pop FOR pop]; [fitnessn FROM pop FOR pop];
                 [parametron FROM pop FOR pop]
         from.w2 ? [cromn FROM (2*pop) FOR pop]; [fitnessn FROM (2*pop) FOR pop];
                 [parametron FROM (2*pop) FOR pop]
       to.monitor ! [cromn FROM 0 FOR popsize/2];
```

```
            [fitnessn FROM 0 FOR popsize/2];
            [parametron FROM 0 FOR popsize/2]
:
```

Si los procesos esclavos no presentan ramificaciones, se tiene el siguiente código:

```
-- ALGORITMO GENETICO ESCLAVO. MODELO FARMING.

... Include's
... Use's

PROC ga5(CHAN OF ANY from.monitor, to.monitor)

  ... Declaracion de Variables

 SEQ
   going:=TRUE
   primr:=TRUE
   WHILE going
     SEQ
       IF
         primr
           SEQ
             from.monitor ? [pool FROM 0 FOR popsize]; xx; seed; maxgen
             primr:=FALSE
         TRUE
           from.monitor ? [pool FROM 0 FOR popsize]; maxgen
       IF
         maxgen <= 0
           going:=FALSE
         TRUE
           SKIP
       ... Cruzamiento
         ... Mutación
       to.monitor ! [crom FROM 0 FOR popsize];
                  [fitness FROM 0 FOR popsize];
                  [parametro FROM 0 FOR popsize]
:
```

El archivo de configuración para el AGP modelo farming es el siguiente:

```
-- ARCHIVO DE CONFIGURACION. MODELO FARMING.
VAL K IS 1024: ·
VAL M IS K*K:
NODE      gac.p:
[4]NODE   ga5.p:
[2]NODE   gaw.p:
ARC       hostlink:

NETWORK farming
  DO
    SET gac.p(type, memsize:="T805",8*M)
```

```
    DO i=0 FOR 4
      SET ga5.p[i](type, memsize:="T805",8*M)
    DO i=0 FOR 2
      SET gaw.p[i](type, memsize:="T805",8*M)
    CONNECT gac.p[link][1] TO HOST WITH hostlink
    CONNECT gac.p[link][2] TO gaw.p[0][link][1]
    CONNECT gac.p[link][3] TO gaw.p[1][link][0]
    CONNECT gaw.p[0][link][0] TO ga5.p[0][link][3]
    CONNECT gaw.p[0][link][3] TO ga5.p[1][link][0]
    CONNECT gaw.p[1][link][2] TO ga5.p[2][link][1]
    CONNECT gaw.p[1][link][3] TO ga5.p[3][link][0]
:

-- mapping
NODE      gac.l:
[4]NODE   ga5.l:
[2]NODE   gaw.l:
MAPPING
  DO
    MAP gac.l ONTO gac.p
    DO i=0 FOR 2
      MAP gaw.l[i] ONTO gaw.p[i]
    DO i=0 FOR 4
      MAP ga5.l[i] ONTO ga5.p[i]
:

--SW
#INCLUDE    "hostio.inc"
#INCLUDE    "protof.inc"
#INCLUDE    "linkaddr.inc"
#USE        "gac.lku"
#USE        "ga5.lku"
#USE        "gaw.lku"

CONFIG
  CHAN OF SP fs:
  CHAN OF SP ts:
  PLACE fs, ts ON hostlink:
  [2]CHAN OF ANY to.worker, from.worker:
  [4]CHAN OF ANY to.nt, from.nt:
  PLACED PAR
    PROCESSOR gac.l
      gac(fs, ts, from.worker[0], from.worker[1], to.worker[0], to.worker[1])
    PAR i=0 FOR 2
      PROCESSOR gaw.l[i]
        gaw(to.worker[i], from.worker[i], from.nt[2*i],from.nt[(2*i)+1],
            to.nt[2*i], to.nt[(2*i)+1])
    PAR i=0 FOR 4
      PROCESSOR ga5.l[i]
        ga5(to.nt[i], from.nt[i])
```