UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

ESTABILIZACION Y OBSERVACION SIMULTANEAS UTILIZANDO ESTRATEGIAS EVOLUTIVAS.

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A :

STALIN MUÑOZ GUTIERREZ

DIRECTOR DE TESIS: DR. GUILLERMO FERNANDEZ ANAYA.

CIUDAD UNIVERSITARIA. 1998.

A Guadalupe y Aurelio mi amor, sobre estas páginas.

...; ...

Entre los demasiados textos que inundan de palabras y símbolos las bibliotecas de nuestras universidades. Se reserva un lugar oscuro, casi mítico, para el almacenamiento de cierto tipo de ellos, cuya existencia sólo conoce el autor (en el mejor de los casos). Podemos intuir que tal característica se debe, no al contenido de los textos, sino más bien a la clase a la que pertenecen. Y aunque este texto, perteneciente a esta categoría no ha de ser excepción. Quiero expresar mi mas completo agradecimiento a quienes lo hicieron posible.

Gracias a **Guillermo**, **Rodolfo** y **Walterio** amigos excelentes y coautores de este trabajo.

Índice General

Capítulo 1

Introducción

En muchas áreas de la ingeniería, disciplinas científicas y en actividades cotidianas nos encontramos frecuentemente con problemas que requieren ser resueltos utilizando los recursos disponibles de la mejor manera. El proceso que nos permite asignar esos recursos de la "mejor manera" es conocido como proceso de *optimización*. En términos generales este proceso de optimización está centrado en optimización de funciones, es decir; dada una función f multidimensional, el problema de optimización de f puede verse como el problema de encontrar un subconjunto S del dominio de la función tal que no existe elemento del complemento de S para el que el valor de la función f es menor que el valor de la función f evaluada en cualquier elemento de S. Dada la definición anterior se hace necesario que el codominio de la función f sea un conjunto ordenado, es decir un conjunto en el que este definida la relación "menor que". En la terminología estándar para el área de optimización a la función f se le conoce como función objetivo y a los elementos del dominio de la función se les llama variables objeto, los elementos de S son llamados mínimos de f. Tipicamente: $f : \Re^n \rightarrow \Re$ donde $\Re$ son los números reales.

Esta definición tan general nos permite expresar muchos problemas de ingeniería como problemas de optimización de funciones, por lo que es útil disponer de algún método u algoritmo que pueda resolverlo sin pedir más restricciones que la definición de los dominios de las variables objeto y definición de la función objetivo. Desafortunadamente un método general no existe, lo que justifica la investigación encaminada a encontrar métodos cada vez más generales que nos auxilien en la solución de dichos problemas.

Existen muchos algoritmos de optimización de funciones. Sin embargo, la mayoria de ellos requiere de condiciones fuertes en la función objetivo para que puedan ser aplicados, entre ellas linealidad, derivabilidad o un número no muy grande de variables. Acontinuación se presenta una descripción de algunos algoritmos, para más detalles consultar [27].

El algoritmo más viejo y quiza más conocido es el *método de Cauchy*, desarrollado en 1847, conocido también como "steepest ascent". Éste se basa en lo siguiente: si F es la función que se desea optimizar, esta función es diferenciable y se toma un pequeño incremento de las variables, y se toman los primeros terminos de la serie de Taylor en el punto $(x_1, x_2, ..., x_n)$, entonces:

$$F(x_1+\delta_1, x_2+\delta_2, ..., x_n+\delta_n) - F(x_1, x_2, ..., x_n) \approx \delta_1 \frac{\partial F}{\partial x_1} + \delta_2 \frac{\partial F}{\partial x_2} + \cdots + \delta_n \frac{\partial F}{\partial x_n}$$

$$= \sqrt{\delta_1^2 + \delta_2^2 + \cdots + \delta_n^2} \sqrt{\left(\frac{\partial F}{\partial x_1}\right)^2 + \left(\frac{\partial F}{\partial x_2}\right)^2 + \cdots + \left(\frac{\partial F}{\partial x_n}\right)^2} \cos\theta$$

θ es el ángulo entre el paso $(\delta_1, \delta_2, .., \delta_n)$ y el vector gradiente en $(x_1, x_2, ..., x_n)$.

Diferentes direcciones de los pasos dan diferentes ángulos y la ecuación anterior sugiere que la dirección de pendiente máxima ocurre cuando $\cos\theta$ es igual a uno. Entonces se elige un paso desde el punto $(x_1, x_2, ..., x_n)$ en dirección de $(g_1, g_2, ..., g_n)$, donde:

$$g_i(x_1, x_2, ..., x_n) = \frac{\partial F(x_1, x_2, ..., x_n)}{\partial x_i}, i = 1, 2, ..., n$$

La longitud de este paso se calcula para maximizar el nuevo valor de la función objetivo. Entonces se remplaza el estimado de la posición del óptimo en $(x_1, x_2, ..., x_n)$ por el estimado $(x_1 + \lambda^* g_1, x_2 + \lambda^* g_2, ..., x_n + \lambda^* g_n)$ donde λ^* es el valor de λ que maximiza la función de una variable:

$$\phi(\lambda) = F(x_1 + \lambda g_1, x_2 + \lambda g_2, ..., x_n + \lambda g_n)$$

Muchos otros algoritmos de optimización se basan en este principio. A partir de una posición estimada $(x_1, x_2, ..., x_n)$ calculan una dirección y cambian el punto estimado por un múltiplo de la dirección, que se calcula para maximizar el cambio en esta dirección. Este método requiere que se resuelva el problema de maximización de la función de una variable $\phi(\lambda)$.

Otro de los métodos más utilizados es el *método de Newton-Raphson* que considera el primer y segundo termino en la serie de Taylor:

$$F(x_1+\delta_1, x_2+\delta_2, ..., x_n+\delta_n) \approx F(x_1, x_2, ..., x_n) + \sum_{i=1}^{n} \delta_i g_i + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \delta_i G_{ij} \delta_j$$

con:

$$G_{ij} = \frac{\partial^2 F(x_1, x_2, ..., x_n)}{\partial x_i x_j}$$

Un punto estimado $(x_1, x_2, ..., x_n)$ se cambia por $(x_1+\delta_1, x_2+\delta_2, ..., x_n+\delta_n)$, donde δ_i se definen como:

$$\delta_i = \sum_{j=1}^{n} (G^{-1})_{ij} g_j \ , \ i = 1, 2, .., n.$$

Otros métodos son los métodos de busqueda directos, en donde no se requiere del calculo de las derivadas de la función. Uno de los más utilizados es el *método de Rosenbrock*. En este método el punto estimado $(x_1, x_2, ..., x_n)$ es cambiado por $\{(x_1, x_2, ..., x_n) + \lambda_1 \mathbf{d}_1\}$, este estimado, a su vez, es cambiado por $\{(x_1, x_2, ..., x_n) + \lambda_1 \mathbf{d}_1 + \lambda_2 \mathbf{d}_2\}$ y así sucesivamente hasta completar la iteración donde el punto estimado se cambia por $(x_1, x_2, ..., x_n) + \sum_{t=1}^{n} \lambda_t \mathbf{d}_t$, siendo $\mathbf{d}_t$ el t-ésimo vector de coordenadas del espacio de búsqueda, antes de comenzar con la nueva iteración la primera dirección de búsqueda es remplazada por $\mathbf{d}_1^* = \sum_{t=1}^{n} \lambda_t \mathbf{d}_t$, Las demás direcciones de busqueda se obtienen por un proceso de ortogonalización. Este proceso se repite iterativamente.

El método de *Hooke y Jeeves* es mejor que el método de Rosenbrock. Este método se basa en dos pasos: un movimiento patrón y un movimiento de exploración. El movimiento patrón se aplica primero y cambia el estimado $\left(x_1^{(k)}, x_2^{(k)}, ..., x_n^{(k)}\right)$ de la posición de la solución por el movimiento total hecho en la última iteración (excepto que en la primera iteración no hay movimiento patrón). El punto resultante es $(y_1, y_2, ..., y_n)$. Desde este punto se hace un movimiento exploratorio, el cual representa un ajuste fino del valor de las variables. Se hacen pequeños cambios alrededor de todas las direcciones coordenadas. El punto resultante es $(z_1, z_2, ..., z_n)$. Si $F(z_1, z_2, ..., z_n)$ es más grande que $F\left(x_1^{(k)}, x_2^{(k)}, ..., x_n^{(k)}\right)$, entonces $(z_1, z_2, ..., z_n)$ se convierte en la nueva aproximación del puntro óptimo para la proxima iteración. Si esto no pasa la iteración se trata como una falla y se hace un nuevo movimiento exploratorio desde $\left(x_1^{(k)}, x_2^{(k)}, ..., x_n^{(k)}\right)$. Una modificación de este método es el método de Baer.

Existen además los métodos de gradiente conjugado, que son ampliamente utilizados. El más conocido y uno de los más exitosos es el método de *Davidon*. Para hacer una breve descripción debe recordarse que en el método de Cauchy, las direcciones de búsqueda están dadas por

$$\delta_i = \sum_{j=1}^{n} I_{ij} g_j \quad i = 1, 2, ..., n$$

donde Ies la matriz unitaria y g_j es la j-ésima componenete del gradiente de la función objetivo. La matriz unitaria puede ser sustituida por cualquier matriz definida positiva, obteniéndose otra dirección de búsqueda que tiene la propiedad de que su ángulo con el vector gradiente es menor que 90°. Si el gradiente no es cero, se obtendra un incremento de la función objetivo en esta dirección. En la iteración k-ésima el método de Davidon cambia el estimado $\left(x_1^{(k)}, x_2^{(k)}, ..., x_n^{(k)}\right)$ por el estimado $\left(x_1^{(k+1)}, x_2^{(k+1)}, ..., x_n^{(k+1)}\right)$, buscando por el máximo de la función en la dirección que tienen los componentes:

$$\delta_i^{(k)} = \sum_{j=1}^{n} H_{ij}^{(k)} g_j^{(k)} \quad i = 1, 2, ..., n$$

$H_{ij}^{(k)}$ es una matriz definida positiva, y se elige con la intención de mejorar la convergencia del método. El método sólo usa valores y derivadas de la función objetivo. Además se suma un término de corrección que depende de dos factores:

$$\sigma^{(k)} = \left(x_1^{(k+1)}, x_2^{(k+1)}, ..., x_n^{(k+1)}\right) - \left(x_1^{(k)}, x_2^{(k)}, ..., x_n^{(k)}\right),$$
$$\gamma^{(k)} = \left(g_1^{(k+1)}, g_2^{(k+1)}, ..., g_n^{(k+1)}\right) - \left(g_1^{(k)}, g_2^{(k)}, ..., g_n^{(k)}\right)$$

de acuerdo a:

$$H^{(k+1)} = H^{(k)} - \frac{\sigma^{(k)}\left(\sigma^{(k)}\right)^T}{\left(\sigma^{(k)}\right)^T \gamma^{(k)}} - \frac{H^{(k)}\gamma^{(k)}\left(\gamma^{(k)}\right)^T H^{(k)}}{\left(\gamma^{(k)}\right)^T H^{(k)} \gamma^{(k)}}$$

este método, usualmente, converge rápidamente.

Existe un conjunto muy grande de algoritmos de optimización que se suman a los anteriormente descritos, pero en su mayoría todos suponen ciertas características que debe cumplir la función a optimizar. Además, el número de variables que pueden manejar dependen fuertemente de la función objetivo, y lo principal, la mayoría de ellos no tienen un buen desempeño si los puntos estimados inicialmente se encuentran lejos del punto óptimo.

El objetivo de este trabajo, resultado de mi colaboración con el Dr. Guillermo Fernández, Rodolfo A. Sánchez y Walterio W. Mayol, es presentar una técnica poco conocida de optimización llamada computación evolutiva, mostrando su utilidad en dos problemas difíciles de control y su fácil extensión a muchos otros problemas en esta misma área. No se presentará aquí una

comparación de los algoritmos clásicos de optimización respecto de las técnicas de computación evolutiva, porque escapa al interés y a los objetivos planteados para esta tesis, pero es válido mencionar que, a la luz de los resultados que expondremos, resultan en una herramienta utilísima y muy general aplicable a muchos problemas de ingeniería.

El problema de estabilización fuerte simultánea es un problema importante en el área de control robusto, donde existen verdaderamente pocos algoritmos que lo resuelvan y para el problema general no existe ninguno. Aquí se presenta una aproximación usando estrategias evolutivas [29][?][12][31]. El problema de observación simultánea [34] es un problema interesante también, en el que se desea encontrar un observador para un conjunto finito de plantas que observe simultáneamente a estas plantas. Este problema es un problema no resuelto cuando se tienen más de dos plantas. Abordaremos este problema utilizando estrategias evolutivas y mostraremos cómo se puede transformar un problema que no era originalmente de optimización en un problema que sí lo es.

Este trabajo tiene la siguiente organización: en el capítulo 2 se presenta una revisión de las estrategias evolutivas, se explican los diferentes tipos y cómo puede diseñarse una función objetivo. En el capítulo 3 se presenta el problema de estabilización fuerte simultánea, cómo se aborda usando estos algoritmos y se dan ejemplos de solución a problemas interesantes. En el capitulo 4 se presenta el problema de observación simultánea con la aproximación de usar estrategias evolutivas como herramienta de solución y se dan los primeros ejemplos existentes para más de dos plantas. En el capítulo 5 se presentan algunas conclusiones importantes de este trabajo. En el apéndice A se podrán encontrar los programas en Scilab para resolver ambos problemas y un programa en Matlab para el problema de estabilización fuerte; finalmente se presenta la descripción de la estrategia simplificada alternativa para el problema de estabilización fuerte simultánea en el apéndice B.

Capítulo 2

Estrategias evolutivas

2.1 Introducción

El área de computación evolutiva nace de la observación de que en la naturaleza se dan de manera inherente procesos de optimización. De los estudios sobre la evolución de las especies surgen algoritmos que análogamente al proceso de evolución encuentran soluciones a problemas específicos. Para observar lo anterior consideremos el siguiente ejercicio de imaginación:

Dado un ecosistema descrito por un conjunto de propiedades o atributos, consideremos, para simplificar, el estudio de una sola especie de animales que lo habitan. La naturaleza ha definido algunas operaciones con los individuos de la población de esta especie, una de las más importantes es una manera de reproducción por medio de la cual un individuo procrea otros individuos de la misma especie. Estos "hijos" del individuo se forman de un proceso de cruza en el que de información de dos individuos "padres" nacen otros más, que no sólo conservan características de ellos, sino que por medio de otro proceso (la mutación) pueden tener cualidades diferentes. Estos hijos pueden ser o no aptos para sobrevivir en el ecosistema. Los más aptos tendrán la oportunidad a sus vez de reproducirse y de heredar características a sus hijos. Los menos aptos estarán en peligro de morir antes de heredar sus características a otro individuo. La analogía que puede observarse con el proceso de optimización se da si consideramos al ecosistema como la función objetivo de nuestro problema a resolver y a los individuos de la especie como las variables objeto del mismo, Todo individuo que sobrevive en el ecosistema puede considerarse una solución al problema de "sobrevivir en el ecosistema". Más aún, el que un individuo sobreviva en el ecosistema requiere que él tenga un conjunto de características particulares que le permitan esto, por

lo que puede verse al individuo como una solución al problema de "encontrar un individuo que cumpla con ciertas características". Aquí el ecosistema es el que se encarga de exigir del individuo dichas características.

De esta visión simplificada de un proceso de evolución son extrapoladas las operaciones de *cruza*, *mutación* y *supervivencia* de los individuos más aptos, aplicándolas a individuos que pueden ser elementos constitutivos de la solución de un problema.

Existen actualmente tres áreas destacadas en computación evolutiva que buscan resolver problemas de optimización,estas son las de *algoritmos genéticos*, la *programación evolutiva* y las *estrategias evolutivas*. Un estudio más profundo de estas técnicas, así como un estudio comparativo puede encontrarse en [2]. La descripción de estas técnicas es omitida en este trabajo que sólo se ocupará de la tercera de ellas: las estrategias evolutivas.

2.2 Descripción de las estrategias evolutivas

La idea principal del funcionamiento de las estrategias evolutivas consiste en utilizar un conjunto de reglas básicas (aplicadas a una población) que tienen analogía con las reglas de evolución biológica. Los principales elementos y reglas son:

- *población:* conjunto de individuos, quienes inicialmente tienen valores aleatorios.
- *individuo:* vector real. Cada individuo de la población es una solución en potencia del problema a resolver.
- *mutación:* modificación aleatoria en alguna parte de la información que conforma al individuo de la población.
- *recombinación:* mezcla de los elementos que componen dos o más individuos.
- *evaluación:* asignación de aptitud a cada individuo de la población por medio de la función objetivo, con el propósito de determinar su eficacia para resolver cierto problema.
- *codificación:* transforma los elementos del dominio del problema a resolver a un vector real que es el individuo.

- *decodificación:* transformación que se aplica a cada individuo de la población para poder medir su aptitud.

- *selección:* separación de los individuos más aptos de los menos aptos que existen en la población.

Existen dos tipos principales de estrategias evolutivas, la estrategia $(\mu, \lambda) - ES$ y la estrategia $(\mu + \lambda) - ES$. La diferencia entre estos tipos de estrategias es que en la estrategia $(\mu, \lambda) - ES$ se tienen poblaciones de μ padres los cuales perecen (no pueden sobrevivir a la siguiente iteración) quedando sólo λ hijos de los que se seleccionan otros μ para la siguiente iteación, es decir ningún individuo puede sobrevivir por siempre y el proceso de selección se realiza con los nuevos descendientes únicamente. En cambio en la estrategia $(\mu + \lambda) - ES$ tanto los padres como los hijos participan del proceso de selección permitiendo a un individuo permanecer en las nuevas generaciones de manera indefinida mientras los hijos que se generan de la población no sean más aptos que él.

2.3 Definiciones y notación

Para simplificar la descripción de las estrategias evolutivas nos auxiliaremos de las siguientes definiciones:

Definición 2.1 Sea $A = [a_{ij}]$ una matriz de $c \times d$ elementos. Definimos el operador submatriz ${}^{k}_{n}sub^{l}_{m} : \Re^{c \times d} \rightarrow \Re^{(l-k+1)\times(m-n+1)}$ tal que: ${}^{k}_{n}sub^{l}_{m}(A) = B = [b_{qr}]$ donde: $b_{qr} = a_{q+k-1\ r+n-1}$; $1 \leq q \leq l-k+1, 1 \leq r \leq m-n+1$.

Ejemplo 2.1 Sea la matriz $A = \begin{bmatrix} 1 & 6 \\ 5 & 4 \end{bmatrix}$, el conjunto Φ formado con todas las submatrices de A, de acuerdo a la definición anterior esta dado por:

$$\Phi = \left\{ \begin{array}{c} {}^{1}_{1}sub^{1}_{1}(A), {}^{1}_{2}sub^{1}_{2}(A), {}^{2}_{1}sub^{2}_{1}(A), {}^{2}_{2}sub^{2}_{2}(A), \\ {}^{1}_{1}sub^{2}_{1}(A), {}^{1}_{2}sub^{2}_{2}(A), {}^{1}_{1}sub^{1}_{2}(A), {}^{2}_{1}sub^{2}_{2}(A), {}^{1}_{1}sub^{2}_{2}(A) \end{array} \right\}$$

$$\Phi = \left\{ \begin{array}{c} [1], [6], [5], [4], \\ \left[\begin{array}{c} 1 \\ 5 \end{array}\right], \left[\begin{array}{c} 6 \\ 4 \end{array}\right], \left[\begin{array}{cc} 1 & 6 \end{array}\right], \left[\begin{array}{cc} 5 & 4 \end{array}\right], \left[\begin{array}{cc} 1 & 6 \\ 5 & 4 \end{array}\right] \end{array} \right\}$$

Definición 2.2 Sean las matrices $A = [a_{ij}] \in \Re^{n \times m}$ y $B = [b_{ij}] \in \{1, 2, ..., n\}^{l \times m}$. La sustitución columna de B (matriz de índices) en A denotada por A_B es: $A_B = [c_{ij}] : c_{ij} = a_{b_{ij} j}$

Ejemplo 2.2 Sean as matrices $A = \left[\begin{array}{cc} 1 & 1 \\ 2 & 2 \end{array}\right]$ y $B = \left[\begin{array}{cc} 2 & 1 \\ 1 & 2 \end{array}\right]$ entonces de acuerdo a la definición anterior las sustituciones columna de B en A y de A en B son respectivamente:

$$A_B = \left[\begin{array}{cc} A_{21} & A_{11} \\ A_{12} & A_{22} \end{array}\right] = \left[\begin{array}{cc} 2 & 1 \\ 1 & 2 \end{array}\right] \text{ y } B_A = \left[\begin{array}{cc} B_{11} & B_{12} \\ B_{21} & B_{22} \end{array}\right] = \left[\begin{array}{cc} 2 & 1 \\ 1 & 2 \end{array}\right]$$

Nótese que en general $A_B \neq B_A$ y que si existe A_B no necesariamente existe B_A. Además las matrices no tienen que tener el mismo número de filas. Después de una operación de sustitución columna, la matriz resultado tendrá las dimensiones de la matriz de índices y todos los elementos de esta matriz serán elementos de la matriz a la que se le aplica la sustitución.

Definición 2.3 Sea $x \in \Re^n$. Se define el operador ordenamiento $ord : \Re^n \rightarrow \Re^{n \times 2}$ como: $ord(x) = \left[\begin{array}{cc} y & z \end{array}\right]$ donde: $y_k = x_p$, $p = z_k$, $q = z_{k+1}$, $y_k \leq y_{k+1}$, $p < q$ si $x_p = x_q$.

Ejemplo 2.3 Sea el vector $x = \left[\begin{array}{cccccc} 6 & 2 & 5 & 3 & 11 & 3 \end{array}\right]^T$ el ordenamiento de x, dada la definición anterior es:

$$ord(x) = \left[\begin{array}{cc} 2 & 2 \\ 3 & 4 \\ 3 & 6 \\ 5 & 3 \\ 6 & 1 \\ 11 & 5 \end{array}\right]$$

del que se observa fácilmente que la primera columna es un vector ordenado en forma creciente, mientras que la segunda columna indica el índice correspondiente de cada elemento en el vector original x.

2.4 Descripción formal

2.4.1 Estrategia $(\mu + \lambda) - ES$ con mutaciones simples.

La descripción de una estrategia evolutiva puede hacerse utilizando n-tuplas. Tales descripciones son detalladas ampliamente en Schwefel y Rudolph [31] o en Hoffmeister y Bäck [12]. A continuación mostramos la estrategia evolutiva $(\mu + \lambda) - ES$ con mutaciones simples.

$$(\mu + \lambda) - ES = \left(P^0, \mu, \lambda, r, m, s, f, T, \tau, \tau_0\right)$$

donde:

$P^0 = [\ X^0 \quad \sigma^0\ ] \in \Re^{\mu \times 2n}$	población
$\mu \in \mathbf{N}$	número de padres
$\lambda \in \mathbf{N}\ ;\ \lambda \geq \mu$	número de hijos
$r : \Re^{\mu \times 2n} \rightarrow \Re^{(\mu+\lambda) \times 2n}$	operador de recombinación
$m : \Re^{(\mu+\lambda) \times 2n} \rightarrow \Re^{(\mu+\lambda) \times 2n}$	operador de mutación
$s : \Re^{(\mu+\lambda) \times 2n} \rightarrow \Re^{\mu \times 2n}$	operador de selección
$f : \Re^n \rightarrow \Re$	función objetivo
$T : \Re^{\mu \times 2n} \rightarrow \{0, 1\}$	criterio de terminación
τ, τ_0	parámetros de control de tamaño de paso

La población se compone de lo siguiente:

$X^t = [(x_1^t)^T, ..., (x\mu^t)^T]^T \in \Re^{\mu \times n}$	matriz de variables objeto
$\sigma = [(\delta_1^t)^T, ..., (\delta\mu^t)^T]^T \in \Re^{\mu \times n}$	tamaños de paso
$a_i^t = [x_i^t \quad \delta_i^t] \in \Re^n$	individuo

El superíndice t denota la iteración en la que se encuentra el algoritmo, el individuo i-ésimo de la población es un vector x_i^t con las variables objeto y un vector δ_i^t de tamaños de paso que regula la mutación que se aplica a los individuos.

El proceso de recombinación permite construir nuevos individuos con información de los padres que forman la población. El operador r se aplica a la población de la siguiente forma:

$$r\left(P^t\right) = \begin{bmatrix} P^t \\ P'^t \end{bmatrix} = \begin{bmatrix} X^t & \sigma^t \\ X'^t & \sigma'^t \end{bmatrix}$$

$P'^t = \begin{bmatrix} X^t_{\xi^t} & \sigma^t_{\xi^t} \end{bmatrix}$, donde ξ^t es una matriz con elementos aleatorios y distribución uniforme sobre $\{1,2,...,\mu\}^{\lambda\times n}$. Aquí hemos aplicado la sustitución columna de ξ^t en X^t y de ξ^t en σ^t para obtener P'^t que se identificaría con la población de hijos.

La mutación m se aplica de la siguiente forma:

$$\widehat{P^t} = \begin{bmatrix} \widehat{X^t} & \widehat{\sigma^t} \end{bmatrix} = m\left(\begin{bmatrix} X^t & \sigma^t \\ X'^t & \sigma'^t \end{bmatrix}\right) = \begin{bmatrix} X^t & \sigma^t \\ X''^t & \sigma''^t \end{bmatrix}$$

con:

$$X''^t = X'^t + N^{\lambda\times n}\left(\sigma''^t\right), \qquad \sigma''^t = \sigma'^t \circ \exp\left(z^t + z_0^t\right)$$
$$z^t \in N^{\lambda\times n}(\tau), \quad z_0^t = \begin{bmatrix} z_{0_1}^t & \dots & z_{0\lambda}^t \end{bmatrix}^T, \quad z_{0_i}^t = \varphi_i^t V$$
$$\varphi_i^t \in N^{1\times 1}(\tau_0), \quad V \in \{1\}^n$$
$$\tau = \frac{1}{2n^{\frac{1}{4}}}, \quad \tau_0 = \frac{1}{\sqrt{2n}}$$

donde $N^{m\times n}(\psi)$ se refiere a la matriz de tamaño $n\times m$ cuyos elementos independientes i,j tienen distribución normal con media cero y desviación estándar ψ_{ij} . Las operaciones exponenciación exp (función exponencial) y multiplicación o se realizan elemento a elemento. Además se observa que sólo los hijos presentan modificaciones aleatorias por la mutación, conservándose intacta la población original de padres.

Una vez que se ha aplicado la mutación m, se mide la aptitud de cada uno de los individuos. Esto se hace con la función objetivo f, y se construye un vector de aptitudes F^t que es de la forma siguiente:

$$F^t = \begin{bmatrix} f\left({}_1^1 sub_n^1(\widehat{X^t})\right) & f\left({}_1^2 sub_n^2(\widehat{X^t})\right) & \dots & f\left({}_1^{\mu+\lambda} sub_n^{\mu+\lambda}(\widehat{X^t})\right) \end{bmatrix}^T$$

Aplicando un ordenamiento a F^t se obtiene $G^t = ord\left(F^t\right)$ y el vector $H^t = \left[{}_2^1 sub_{2\mu}(G)\right]$ con el que se construye $I^t = [H^t, H^t, ..., H^t] \in \Re^{\mu\times 2n}$. I^t permitirá extraer a los individuos más

aptos de la población con el operador de selección.

El operador de selección s se encarga de hacer una selección de los mejores individuos de la población. Son estos los que pasan a la siguiente generación, este operador puede expresarse como:

$$P^{t+1} = s\left(m\left(r\left(P^t\right)\right)\right) = s\left(\widehat{P^t}\right) = \widehat{P^t}_{I^t}$$

Esto resulta de la sustitución columna de I^t en $\widehat{P^t}$.

Una vez aplicada la selección se verifica el criterio de terminación T, si se tiene que $T\left(P^{t+1}\right) = 1$, el algoritmo termina y la mejor solución al problema se encuentra en $w = \left[{}^1_1 sub^1_n\left(P^{t+1}\right)\right]$.

2.4.2 Estrategia $(\mu, \lambda) - ES$ con mutaciones simples.

A continuación mostramos la estrategia evolutiva $(\mu, \lambda) - ES$ con mutaciones simples, que es muy parecida a la estrategia $(\mu + \lambda) - ES$, salvo porque en ésta los padres solo pueden sobrevivir una iteración.

$$(\mu, \lambda) - ES = \left(P^0, \mu, \lambda, r, m, s, , f, T, \tau, \tau_0\right)$$

donde:

$P^0 = [\ X^0 \quad \sigma^0\ ] \in \Re^{\mu \times 2n}$	población
$\mu \in \mathbf{N}$	número de padres
$\lambda \in \mathbf{N}\ ;\ \lambda \geq \mu$	número de hijos
$r : \Re^{\mu \times 2n} \rightarrow \Re^{\lambda \times 2n}$	operador de recombinación
$m : \Re^{\lambda \times 2n} \rightarrow \Re^{\lambda \times 2n}$	operador de mutación
$s : \Re^{\lambda \times 2n} \rightarrow \Re^{\mu \times 2n}$	operador de selección
$f : \Re^n \rightarrow \Re$	función objetivo
$T : \Re^{\mu \times 2n} \rightarrow \{0, 1\}$	criterio de terminación
τ, τ_0	parámetros de control de tamaño de paso

La población se compone de la misma manera que para la estrategia $(\mu + \lambda) - ES$:

$$X^t = [(x_1^t)^T, ..., (x\mu^t)^T]^T \in \Re^{\mu \times n} \quad \text{matriz de variables objeto}$$
$$\sigma = [(\delta_1^t)^T, ..., (\delta\mu^t)^T]^T \in \Re^{\mu \times n} \quad \text{tamaños de paso}$$
$$a_i^t = [x_i^t \quad \delta_i^t] \in \Re^n \quad \text{individuo}$$

El operador de recombinación ahora se define como:

$$r\left(P^t\right) = \left[P'^t\right] = \left[\begin{array}{cc} X'^t & \sigma'^t \end{array}\right]$$

$P'^t = \left[\begin{array}{cc} X^t_{\xi^t} & \sigma^t_{\xi^t} \end{array}\right]$, donde ξ^t es una matriz con elementos aleatorios y distribución uniforme sobre $\{1, 2, ..., \mu\}^{\lambda \times n}$. Obsérvese que tras la recombinación sólo han quedado los hijos. Los padres ya no participan en adelante en el algoritmo.

La mutación m se aplica de la siguiente forma:

$$\widehat{P^t} = \left[\begin{array}{cc} \widehat{X^t} & \widehat{\sigma^t} \end{array}\right] = m\left(\left[\begin{array}{cc} X'^t & \sigma'^t \end{array}\right]\right) = \left[\begin{array}{cc} X''^t & \sigma''^t \end{array}\right]$$

con:

$$X''^t = X'^t + N^{\lambda \times n}(\sigma''^t), \qquad \sigma''^t = \sigma'^t \circ \exp(z^t + z_0^t)$$
$$z^t \in N^{\lambda \times n}(\tau), \quad z_0^t = \left[\begin{array}{ccc} z_{0_1}^t & \cdots & z_{0_\lambda}^t \end{array}\right]^T, \quad z_{0_i}^t = \varphi_i^t V$$
$$\varphi_i^t \in N^{1 \times 1}(\tau_0), \quad V \in \{1\}^n$$
$$\tau = \frac{1}{2n^{\frac{1}{4}}}, \quad \tau_0 = \frac{1}{\sqrt{2n}}$$

donde $N^{m \times n}(\psi)$ se refiere a la matriz de tamaño $n \times m$ cuyos elementos independientes i, j tienen distribución normal con media cero y desviación estándar ψ_{ij}. Las operaciones exponenciación exp y multiplicación $\circ$ se realizan elemento a elemento.

Una vez que se ha aplicado la mutación m, se mide la aptitud de cada uno de los individuos. Esto se hace con la función objetivo f, y se construye un vector de aptitudes F^t, de la misma forma que para la estrategia $(\mu + \lambda) - ES$:

$$F^t = \left[\begin{array}{ccccc} f\left({}_1sub_n^1(\widehat{X^t})\right) & f\left({}_1sub_n^2(\widehat{X^t})\right) & \cdots & f\left({}_1\lambda sub_n\lambda(\widehat{X^t})\right) \end{array}\right]^T$$

Aplicando un ordenamiento a F^t se obtiene $G^t = ord\left(F^t\right)$ y el vector $H^t = \left[\frac{1}{2}sub_2\mu\left(G\right)\right]$ con el que se construye $I^t = \left[H^t, H^t, ..., H^t\right] \in \Re^{\mu \times 2n}$.

El operador de selección s se encarga de hacer una selección de los mejores individuos de la población de hijos, este operador se expresa como:

$$P^{t+1} = s\left(m\left(r\left(P^t\right)\right)\right) = s\left(\widehat{P^t}\right) = \widehat{P^t}_{I^t}$$

Una vez aplicada la selección se verifica el criterio de terminación T. Si se tiene que $T\left(P^{t+1}\right) = 1$, el algoritmo termina y la mejor solución al problema se encuentra en $w = \left[\frac{1}{1}sub_n^1\left(P^{t+1}\right)\right]$.

2.4.3 Características de las estrategias $(\mu + \lambda) - ES$ y $(\mu, \lambda) - ES$

Obsérvese que las descripciones presentadas suponen que el conjunto sobre el que se definen las variables objeto es el conjunto de los números reales y un individuo es un vector n-dimensional sobre este conjunto. Esto es una ventaja sobre los algoritmos genéticos que utilizan cadenas binarias o alfabetos discretos como individuos, dado que resolver problemas con magnitudes cuantitativas en ese caso suponía una codificación especial y en estrategias evolutivas se manejan directamente los números reales. Por lo tanto podremos utilizar estrategias evolutivas a problemas de optimización. que puedan codificar sus soluciones como un vector en $\Re^n$. A pesar de que la estrategia evolutiva descrita anteriormente es aplicable directamente a problemas de minimización, esto no le impide resolver problemas de maximización si se observa que $máx\left(x\right) = mín\left(-x\right)$.

El comportamiento del algoritmo utilizando estrategias $(\mu, \lambda) - ES$ difiere del $(\mu + \lambda) - ES$. En el primero, dado que los padres no pueden sobrevivir a la siguiente generación, la aptitud de los individuos de la población no siempre decrece, pudiendo presentar comportamientos oscilatorios. En cambio en la estrategia $(\mu + \lambda) - ES$ siempre se tiene al mejor individuo en la generación siguiente, por lo cual la mejor aptitud de la población se comporta como una función decreciente respecto de las iteraciones, lo que puede representar una convergencia más rápida, pero con el peligro de caer más fácilmente en mínimos locales que la estrategia $(\mu, \lambda) - ES$.

Las estrategias evolutivas pueden hacer tender algunos de los pasos del individuo a cero,

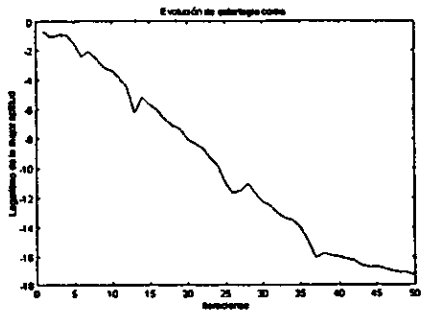

Figura 2-1: Evolución de la mejor aptitud para la estrategia $(\mu, \lambda) - ES$ para el problema de una hiperesfera de cinco variables, 7 padres, 50 hijos, 50 iteraciones.

limitando la exploración sobre alguno de los ejes coordenados. Para disminuir un poco esta dificultad se incorporan un conjunto de parametros llamados ángulos. Los ángulos evolucionan de manera similar a los pasos, efectuando una mutación correlacionada que no resuelve del todo el problema y que requiere de un costo de cómputo mayor.

Ejemplo 2.4 Dada la función objetivo de una esfera n-dimensional de cinco variables se desea obtener el mínimo de la función, esto es : $\min_x[f(x)] = \sum_{i=1}^{5} x_i^2$, utilizando para ello las estrategias $(7,50) - ES$ y $(7+50) - ES$.

En este ejemplo la función objetivo es directamente la función de la hiperesfera. Las variables objeto son vectores en $\Re^5$ donde la codificación puede ser: $\left[\begin{array}{ccccc} x_1 & x_2 & x_3 & x_4 & x_5 \end{array} \right]$. En las figuras 2-1 y 2-2 se ilustra la evolución de la mejor aptitud de ambas estrategias, donde puede verse que para la estrategia $(\mu, \lambda) - ES$ se dan oscilaciones, mientras que la estrategia $(\mu + \lambda) - ES$ es siempre decreciente.

2.5 Función objetivo

La definición de la función objetivo es quizá la parte más importante cuando se utilizan estrategias evolutivas. La función objetivo debe elegirse de tal manera que asigne una aptitud cada vez menor si el vector que evalúa se aproxima a la solución del problema. Éste es el caso en que se trata de encontrar x tal que $f(x)$ es el mínimo.

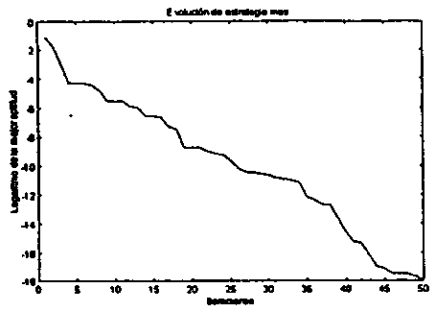

Figura 2-2: Evolución de la mejor aptitud para la estartegia $(\mu + \lambda) - ES$ aplicada a una hiperesfera de cinco variables, 7 padres, 50 hijos y 50 iteraciones.

Las características deseables de una función objetivo es que sea suave (que no tenga discontinuidades o cambios bruscos), que sólo tenga un mínimo (el mínimo global) y que si tiene mínimos locales estos no sean pronunciados. Además es deseable que la función objetivo no asigne a diferentes vectores del dominio de la función la misma aptitud si éstos se encuentran cerca entre sí, esto es que las superficies n-dimensionales descritas por la función objetivo no tengan regiones constantes, debido a que regiones constantes no brindan información al algoritmo para que éste evolucione adecuadamente. En un problema particular la función objetivo puede ser el problema mismo. Tal es el caso del ejemplo presentado en la sección anterior o puede ser una codificación de un problema que originalmente no era un problema de optimización. Para referirnos a estos dos tipos de problemas llamaremos al primero *problema de optimización directo* y al segundo *problema de optimización indirecto*.

Ejemplo 2.5 Se desea resolver el problema de encontrar el mínimo de la función $f(x) = log(x^6 + 1) + 45x^2$

En este caso tan simple de una sola variable la función objetivo es directamente la ecuación $f(x)$, y x es la variable objeto, se trata de un problema de optimización directo.

Ejemplo 2.6 Considérese para el ejemplo anterior que se desea que la variable x además cumpla con la siguiente restricción $30 < x < 45$, entonces existen muchas formas de definir la función objetivo para que esto sea cierto:

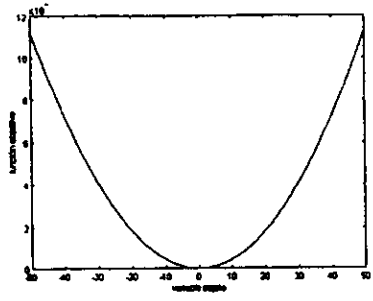

Figura 2-3: Gráfica de la función $f(x)$ para el ejemplo 2.5

$$f_1(x) = \begin{cases} log(x^6+1) + 45x^2 & \text{si} \quad 30 \leq x \leq 45 \\ \infty & \text{si} \quad 30 > x > 45 \end{cases}$$

Esta es la definición más simple de la función objetivo en la cual todos los valores fuera del intervalo son asignados con el valor de ∞, con lo que se elimina la posibilidad de que sean mínimos. Sin embargo no es conveniente porque la función tiene una región importante del espacio de búsqueda constante. Esto se observa en la figura 2-4. Una función más apropiada puede ser la siguiente:

$$f_2(x) = \begin{cases} log(x^6+1) + 45x^2 & \text{si} \quad 30 \leq x \leq 45 \\ K(log(x^6+1) + 45x^2 + 1) & \text{si} \quad 30 > x > 45 \end{cases}$$

donde K es una constante positiva grande, esta constante se dice que penaliza al individuo si no cumple con la restricción de dominio de pertenencia. Si esta constante no se lleva al caso limite (∞), la función conserva la forma de la función original salvo por el desplazamiento que da el término unitario, así que se tendrán resultados mejores con esta función que si se utilizase la primera. Puede observarse en la figura 2-5 que fuera de la región de interés (región de penalización) la función tiene la forma de la función original, por lo cual el algoritmo tiene un mejor desempeño usando esta última.

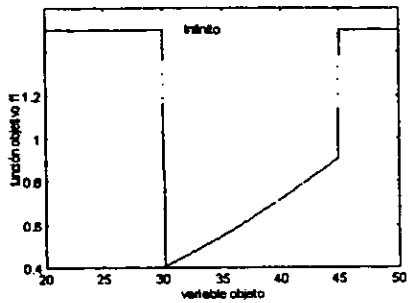

Figura 2-4: Gráfica de la función objetivo f_1 para el ejemplo 2.6

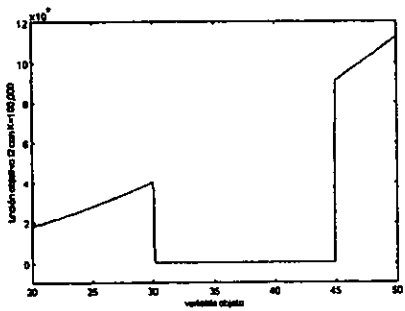

Figura 2-5: Gráfica de la función objetivo f_2 para el ejemplo 2.6

Existen numerosas variantes de definición de función objetivo para el tratamiento de restricciones, se recomienda consultar [21] para mas detalles.

2.6 Conclusiones

Las estratégias evolutivas son algoritmos eurísticos de solución a problemas de optimización de funciones. Se basan en el principio de evolución natural, por lo que se dice que las soluciones evolucionan conforme el algoritmo itera.

Las operaciones en que se basa son: cruza de individuos o recombinación, mutaciones y selección de los individuos más aptos.

Las Estrategias evolutivas se clasifican en dos tipos principales, la estrategia $(\mu, \lambda) - ES$ y la estrategia $(\mu + \lambda) - ES$. Sin embargo aceptan muchas variantes: las hay con mutaciones simples o con mutaciones correlacionadas con varios tipos de recombinación y de selección.

En este capítulo se ha presentado una descripción de las estratégias evolutivas, mostrandose algunas de las consideraciones que deben tenerse en el diseño y aplicacion de las mismas, entre ellas: poder codificar el problema a un vector en $\Re^n$, evitar la definición de funciones con multiples mínimos, y funciones con regiones del espacio constante.

Se presentaron algunos ejemplos, observandose de ellos la simplicidad de la estrategia para su implementación en computadoras de propósito general. En los capítulos siguientes se mostrarán ejemplos concretos de aplicación y se hará explicito el proceso de solución del problema.

Capítulo 3

Estabilización Fuerte Simultánea

3.1 Introducción

El problema de estabilización simultánea por un controlador dinámico lineal para una colección finita de plantas[1], es importante en el área de control robusto. Por ejemplo, cuando se requiere la estabilidad de un sistema operando en diferentes modos, o cuando un modelo linealizado de un sistema no lineal operando en diferentes puntos de operación tiene parámetros variando en el tiempo y se desea estabilizarlo. También cuando ocurren fallas en un sistema y el sistema posterior a cada falla tiene un modelo diferente y se requiere diseñar un controlador que estabilice simultáneamente los sistemas resultantes. Otro caso es el de estabilizar un sistema lineal con parámetros inciertos o variantes en el tiempo, donde se desea diseñar un controlador que estabilice el sistema para varios conjuntos representativos de los parámetros. Finalmente, varios problemas de estabilización de plantas de intervalos en control robusto se reducen a la estabilización simultánea de las plantas "esquinas" resultantes.

La solución al problema de encontrar un algoritmo eficiente y rápido para obtener un controlador que estabilize simultáneamente m plantas SISO lineales invariantes en el tiempo, solo ha podido ser resuelto para el caso de un controlador constante por Howitt y Luus [14] y Paskota et al. [25] entre otros; ellos trabajan en el espacio de estados y dan algoritmos eficientes. Sin embargo para controladores dinámicos estabilizando más de dos plantas, el problema es difícil y

[1] Los preliminares básicos de control que describen una planta y un controlador, así como los conceptos de estabilidad y otros pueden consultarse en [24]

esta abierto como establece Blondel et al. [4], de hecho Blondel [4] probó en su libro, que no es posible decidir racionalmente, cuando un conjunto de tres o más sistemas es simultáneamente estabilizable, es decir es NP-Hard.

Recientemente, Bredemann [5] establece que si los numeradores de las plantas diferencia entre una planta fija y las otras son Hurwitz estables, todas las plantas diferencia son estrictamente propias y tienen el mismo signo de alta frecuencia, entonces son simultáneamente estabilizables. También da condiciones suficientes para la existencia de un controlador estable, que simultáneamente estabilize m plantas SISO usando un argumento H^∞ , pero supone que las plantas satisfacen (parity interlacing property) PIP[2]. Un algoritmo de interpolación de unidades racionales reales y acotadas, es usado para construir los controladores. Además, prueba que si m plantas son simultáneamente estabilizables entonces existe un controlador de grado relativo cero que las estabiliza simultáneamente.

Cuando además se exige que el controlador que estabilice los sistemas sea un controlador estable, este problema se conoce como *estabilización simultánea fuerte*. Recientemente Abdallah et al. [1] presentan nuevas condiciones suficientes para el problema de estabilización simultánea fuerte, da un algoritmo de construcción de controladores, sin embargo las condiciones que pide son fuertes: que todas las plantas tengan los mismos ceros inestables y que sean del mismo grado relativo.

3.2 Planteamiento del problema

El problema de estabilización fuerte simultánea es el siguiente:

Sean $P_1(s) = \frac{n_1(s)}{d_1(s)}, ..., P_m(s) = \frac{n_m(s)}{d_m(s)}$, plantas lineales invariantes en el tiempo SISO y propias, $\Re(s)$ es el anillo de funciones racionales reales y $\Re[s]$ es el anillo de polinomios reales y $\Re H^\infty$ es el anillo de las funciones racionales propias y estables, donde $P_i(s) \in \Re(s)$ y son propias, $n_i(s), d_i(s)$ es la factorización coprima de la planta $P_i(s)$, i.e. $n_i(s), d_i(s) \in \Re H^\infty$ y satisfacen la ecuación:

$$n_i(s)x_i(s) + d_i(s)y_i(s) = 1 \tag{3.1}$$

[2]La propiedad PIP establece que existe un numerto par de polos reales inestables entre cada par de ceros reales inestables..

para $x_i(s)$, $y_i(s) \in \Re H^\infty$ donde $i = 1, 2, ..., m$.

El problema es encontrar un controlador estable:

$$C(s) = \frac{n_c(s)}{d_c(s)} \tag{3.2}$$

tal que:

$$n_i(s)n_c(s) + d_i(s)d_c(s) = 1 \tag{3.3}$$

para $i = 1, 2, ..., m$.

Para el planteamiento del problema en el caso del algoritmo, hacemos lo siguiente:

sean $P_1(s) = \frac{N_1(s)}{D_1(s)}, ..., P_m(s) = \frac{N_m(s)}{D_m(s)}$ las plantas donde $N_i(s), D_i(s) \in \Re[s]$, son polinomios coprimos para cada $i = 1, 2, ..., m$. Ahora se propone un par de polinomios coprimos: $N_c(s), D_c(s) \in \Re[s]$ donde $D_c(s)$ es un polinomio Hurtwitz y:

$C(s) = \frac{N_c(s)}{D_c(s)}$, tal que:

$$N_i(s)N_c(s) + D_i(s)D_c(s) = h_i \; ; \qquad i = 1, 2, ..., m \tag{3.4}$$

los $h_i(s)$ son polinomios estables.

Para que exista un controlador estable que estabilice simultáneamente el conjunto de plantas es necesario que cada planta satisfaga la propiedad PIP.

3.3 Aproximación utilizando Estrategias Evolutivas

En trabajos anteriores Fernández, Muñoz, Sánchez y Mayol [9][22] han presentado un algoritmo utilizando estrategias evolutivas para el problema de estabilización simultánea, en él se presentan ejemplos de hasta ochenta plantas estabilizadas simultáneamente con este algoritmo pero sin pedir la condición fuerte, esto es que el controlador sea estable. Aquí presentaremos nuevos resultados de estabilización fuerte simultánea basados en estos trabajos.

3.3.1 Codificación y decodificación

Si se tienen m plantas $P_i = \frac{N_i}{D_i}$ $\forall i = 1, 2, 3, ...m.$ y un controlador $C = \frac{N_C}{D_C}$ con $N_i, D_i, N_C, D_C \in \Re[s]$ $\forall i$, entonces la estrategia evolutiva puede optimizar iterativamente la población en función de las raíces de los polinomios $h_i = N_i N_C + D_i D_C$, buscándose que todos los valores s_0 tales que $h_i(s_0) = 0$ se localicen en el semiplano complejo izquierdo y que el controlador encontrado sea estable (estabilización fuerte simultánea). Si el algoritmo encuentra un controlador C que estabilice simultáneamente las m plantas entonces tendrá una aptitud igual a *cero* y en cualquier otro caso una aptitud superior, se decide entonces que si el algoritmo encuentra un individuo con aptitud cero, este individuo es una solución al problema.

En la definición de las variables objeto en relación al problema que se aborda existen varias alternativas, tres de ellas son:

1. Las variables objeto son los coeficientes de las funciones racionales (funciones de transferencia) que definen los posibles controladores en el dominio de la frecuencia.

2. Las variables objeto son los polos, ceros y ganancia de alta frecuencia de las funciones de transferencia de los posibles controladores.

3. Las variables objeto son las entradas de matrices que representan al controlador en el espacio de estados.

Si se elige la primera, el dominio de cada una son los números reales y no hay que hacer más consideraciones. En cambio si se elige la segunda el dominio de las variables objeto serán los números complejos, además hay que hacer consideraciones importantes respecto de la cerradura de los polinomios de coeficientes reales, garantizándose que las raíces complejas ocurran en pares conjugados.

La tercera de las posibilidades es fácil para la codificación asociando las entradas de las matrices A,B,C y D con los elementos del vector real n-dimensional. Sin embargo, dados los teoremas bajo los que se diseñará la función objetivo, no es conveniente optar por ésta, a menos que se haga un nuevo planteamiento para esta representación.

Puede verse a simple vista que es más sencillo considerar a las variables objeto como coeficientes, ya que la codificación para estrategias evolutivas se simplifica mucho. Además tiene la

característica de que algunos coeficientes pueden anularse (o tender a cero), permitiendo que conforme avanza el algoritmo se den cambios de grado relativo o de orden del controlador a un orden menor que el orden definido inicialmente. Si no se desea un orden específico esto puede ser una ventaja. Por otro lado, el considerar la segunda codificación involucra más trabajo de programación, pero las superficies descritas por la función objetivo parecen más nobles que en el primer caso, consúltese Fernández et al. [9] para una descripción detallada de este tipo de codificación o véase el apéndice B.

Una vez que hemos definido el dominio de las variables objeto, hay que definir las funciones de codificación y decodificación a un vector en $\Re^n$ con el que se trabaja en las estrategias evolutivas. Para ello definamos lo siguiente:

Definición 3.1 Sea $x \in \Re^n$ definimos la operación pol como $pol(-,s) : \Re^n \rightarrow \Re[s]$ y es tal que:

$$pol\,(x,s) = \sum_{i=0}^{n-1} x_{i+1} s^i$$

Ejemplo 3.1 Sea $x = \begin{bmatrix} 5 & 3 & 1 & 2 \end{bmatrix}$ entonces al aplicar pol a x tenemos:
$pol\,(x,s) = 5 + 3s + s^2 + 2s^3$

Definición 3.2 Sea $x(s) \in \Re[s]$ definimos la operación pol^{-1} como:
$pol^{-1}\,(x(s)) = [y_1, ..., y_n]$ donde y_i es el coeficiente de s^i y $n = \deg x(s)$.

Para la codificación necesitamos que cada individuo sea un controlador posible para nuestro conjunto de plantas, si se trabaja en el dominio de la frecuencia se tienen funciones de transferencia que representan al controlador, así podemos definir la siguiente función:

Definición 3.3 Sea $x \in \Re^n, n$ es un número par, definimos la operación transformación a función racional de orden $\frac{n}{2} - 1$ como:

$$trans\,(x,s) = \frac{pol\left({}_1^{\frac{1}{2}}sub_{\frac{n}{2}}^{1}\,(x), s\right)}{pol\left({}_{\frac{n}{2}+1}^{\frac{1}{2}}sub_{n}^{1}\,(x), s\right)}$$

Esta transformación es nuestra función de decodificación y la función $trans^{-1}$ es la función de codificación. Donde $trans^{-1} : \Re[s] \rightarrow \Re^n$ y se tiene que dada $x(s)$ una función racional de grado relativo cero y $x(s) = \frac{N(s)}{D(s)}$ con $N(s), D(s)$ polinomios coprimos:
$trans^{-1}\,(x(s)) = \begin{bmatrix} V_N & V_D \end{bmatrix}$ donde $V_N = pol^{-1}\,(N(s))\,, V_D = pol^{-1}\,(D(s))\,.$

3.3.2 Función objetivo

Para el diseño de la función objetivo hay que hacer referencia a las condiciones presentadas en la sección de planteamiento del problema de estabilización fuerte simultánea, en ellas se puede observar que para que un controlador estabilize fuertemente a un conjunto finito de plantas es necesario que todos los polinomios de la ecuación 3.4 sean polinomios estables (todas sus raíces en el semiplano complejo izquierdo).

En este caso, nuestro problema no es un problema de optimización explícito. Dado que es del tipo *poi*, se tiene libertad en la elección de la función objetivo, las condiciones que deben presentarse para que el problema sea resuelto son las siguientes:

1. $\exists\ (N_c, D_c \in \Re[s])$ tales que $C(s) = \frac{N_c}{D_c}$
2. $D_c(s_0) = 0 \Longrightarrow R_e\{s_0\} < 0$
3. $\forall h_i$ de la ecuación 3.4 se cumple que $h_i(s_0) = 0 \Longrightarrow R_e\{s_0\} < 0$

la condición 2 es solicitada para que el controlador sea estable, mientras que la condición 3 se pide para que los sistemas sean estabilizados por $C(s)$.

Una función objetivo posible en este caso es la siguiente:

Sea P el conjunto de k plantas a estabilizar, $P = \left\{ \begin{array}{cccc} \frac{N_1}{D1} & \frac{N_2}{D_2} & \ldots & \frac{N_k}{Dk} \end{array} \right\}$. La función objetivo f esta dada por:

$$f(x) = \begin{cases} \sum_{i=1}^{k} \Psi\left(pol\left(\frac{1}{1}sub_{\frac{n}{2}}^{1}(x), s\right) N_i + pol\left(\frac{1}{\frac{n}{2}+1}sub_{n}^{1}(x), s\right) D_i, \varepsilon\right) & \text{si } D_c \text{ es estable} \\ \sum_{i=1}^{k} \frac{\Psi\left(pol\left(\frac{1}{1}sub_{\frac{n}{2}}^{1}(x), s\right) N_i + pol\left(\frac{1}{\frac{n}{2}+1}sub_{n}^{1}(x), s\right) D_i, \varepsilon\right)+1}{\Gamma\left(pol\left(\frac{1}{\frac{n}{2}+1}sub_{n}^{1}(x), K\right)\right)} & \text{en otro caso} \end{cases}$$

de acuerdo a las definiciones siguientes:

Definición 3.4 Si se tiene que $X(s)$ es un polinomio real, éste se expresa por la factorización:

$$X(s) = \lambda \cdot \prod_{i=1}^{\deg(X)} (s - x_i) \quad \text{con } x_i \in \mathbf{C} \text{ y } \lambda \in \Re$$

Haciendo el cambio de variable: $z_i = Re\{x_i\}$, se define Ψ como:

$$\Psi(X(s), \varepsilon) = \sum_{i=1}^{deg(X)} peso(z_i, \varepsilon)$$

donde:

$$peso(z_i, \epsilon) = \begin{cases} \tanh(z_i + \epsilon) & \text{si} \quad z_i \geq 0 \\ 0 & \text{si} \quad z_i < 0 \end{cases}$$

$deg(X)$ es el grado del polinomio $X(s)$. Y ϵ es una constante positiva pequeña.

Definición 3.5 Si se tiene que $X(s)$ es un polinomio real, éste se expresa por la factorización:

$$X(s) = \lambda \cdot \prod_{i=1}^{deg(X)} (s - x_i) \quad \text{con } x_i \in \mathbf{C} \text{ y } \lambda \in \Re$$

Haciendo el cambio de variable: $z_i = Re\{x_i\}$, se define Γ como:

$$\Gamma(X, K) = \begin{cases} K & \text{si} \quad \exists\, i \mid (z_i \geq 0) \\ 1 & \text{si} \quad z_i < 0 \qquad i = 1, 2, ..., deg(X) \end{cases}$$

Para la función objetivo anterior debe elegirse ϵ y K como constantes positivas. K en particular debe ser muy pequeña para que se penalice al individuo cuando el controlador no es estable. En este trabajo se han elegido $K = 1 \times 10^{-9}$, y $\epsilon = 0.1$.

3.4 Ejemplos

En los siguientes ejemplos se ha usado la estrategia $(\mu + \lambda) - ES$, sin embargo la solución de los mismos también se consigue con la estrategia $(\mu, \lambda) - ES$. Se decidió utilizar esta estrategia porque resultó en una convergencia más rápida.

Ejemplo 3.2 En este ejemplo se presenta el uso de la función objetivo de la sección anterior para solucionar un problema de estabilización simultánea fuerte de un conjunto de tres plantas de fase no mínima, todas las plantas de este ejemplo tienen un cero en común y son inestables, este tipo de plantas son tratadas por Abdallah et al. [1], mostramos que las estrategias evolutivas pueden ser útiles en la solución de este problema.

$$P_1 = \frac{s-1}{-3s+1} \quad P_2 \frac{s-1}{-4s+2} \quad P_3 \frac{s-1}{-5s+3}$$

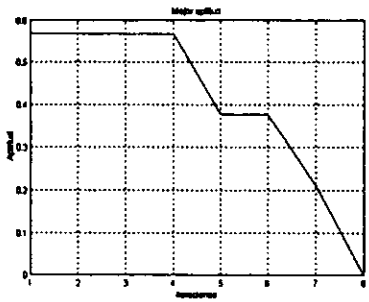

Figura 3-1: Evolución de la mejor aptitud para el ejemplo 3.2

el controlador encontrado es:

$$C(s) = \frac{1.120451 + 2.0585007s + 1.5351784s^2}{.3426178 + .8238771s + .5501619s^2}$$

los parámetros del algoritmo son los siguientes: $\mu = 7$, $\lambda = 50$. Se encontró la solución en tan sólo 7 iteraciones. Ver la figura 3-1

Example 1

Ejemplo 3.3 En el siguiente ejemplo, mostramos como se puede trabajar con ejemplos de fase no mínima que no tengan los ceros inestables comunes, las plantas son:

$$P_1 = \frac{-.75+4s+s^2}{7-2s+2s^2} \quad P_2 = \frac{-1+4s+s^2}{7-2s+s^2} \quad P_3 = \frac{-1+4s+2s^2}{7-3s+s^2} \quad P_4 = \frac{-2+3s+2s^2}{7-5s+3s^2}$$

para este ejemplo el controlador encontrado fue:

$$C(s) = \frac{1.8866286 + .9257070s + .8054580s^2}{.6167782 + .4352455s + .3302904s^2}$$

el cual fue encontrado en dos iteraciones. Usando $\mu = 7$, $\lambda = 50$.

Ejemplo 3.4 En el siguiente ejemplo se presentan 8 plantas en las que existe cambio de signo

de alta frecuencia:

$$P_1 = \frac{-2+s+s^2}{-2-3s+s^2} \quad P_2 = \frac{3+s+10s^2}{-1-5s+s^2} \quad P_3 = \frac{3+20s+10s^2}{-11-5s+s^2} \quad P_4 = \frac{20+13s+s^2}{5-7s+s^2}$$
$$P_5 = \frac{3+10s+s^2}{13-7s+7s} \quad P_6 = \frac{13+5s+s^2}{11-11s+5s^2} \quad P_7 = \frac{-10-15s-s^2}{1-10s+s^2} \quad P_8 = \frac{-11-10s-4s^2}{1-s+11s^2}$$

el controlador encontrado es:

$$C(s) = \frac{1.0725415 + 1.1867118s + 8.4782181s^2 + 2.3423723s^3 + 1.708883s^4}{.0623294 + .1743264s + .4891373s^2 + .8834303s^3 + .1648847s^4}$$

el controlador se encontró en 12 iteraciones usando $\mu = 7$, $\lambda = 49$.

Ejemplo 3.5 En este ejemplo se presentan 100 plantas de segundo grado, las cuales son fuertemente estabilizadas por un controlador encontrado usando estrategias evolutivas, los coeficientes de estas plantas fueron generados aleatoriamente en el intervalo $(0, 1)$, todas las plantas son de fase mínima y con grado relativo cero, pero se invirtió uno de los signos de los coeficientes de los denominadores para que todas fuesen inestables.

Las funciones de transferencia de las plantas son las siguientes:

$$P_1 = \frac{.2113249+.7560439s+.0002211s^2}{-.3875689+.7096427s+.6686618s^2} \qquad P_2 = \frac{.3303271+.6653811s+.6283918s^2}{-.2191974+.6610226s+.4877657s^2}$$

$$P_3 = \frac{.8497452+.6857310s+.8782165s^2}{-.9874265+.7252351s+.4685972s^2} \qquad P_4 = \frac{.0683740+.5608486s+.6623569s^2}{-.6185800+.5101435s+.3562649s^2}$$

$$P_5 = \frac{.7263507+.1985144s+.5442573s^2}{-.1833024+.2824866s+.3442325s^2} \qquad P_6 = \frac{.2320748+.2312237s+.2164633s^2}{-.5313196+.2214915s+.4253308s^2}$$

$$P_7 = \frac{.6833888+.6525135s+.3076091s^2}{-.1505153+.5767575s+.5588058s^2} \qquad P_8 = \frac{.9329616+.2146008s+.312642s^2}{-.9561535+.7048253s+.3304865s^2}$$

$$P_9 = \frac{.3616361+.2922267s+.5664249s^2}{-.6994941+.5607946s+.4157403s^2} \qquad P_{10} = \frac{.4826472+.3321719s+.5935095s^2}{-.2583465+.3628880s+.1595823s^2}$$

$$P_{11} = \frac{.5015342+.4368588s+.2693125s^2}{-.6155266+.3088326s+.3883103s^2} \qquad P_{12} = \frac{.6325745+.4051954s+.9184708s^2}{-.1903011+.9813415s+.9415461s^2}$$

$$P_{13} = \frac{.0437334+.4818509s+.2639556s^2}{-.4607577+.7428482s+.4722979s^2} \qquad P_{14} = \frac{.4148104+.2806498s+.1280058s^2}{-.5220010+.1616391s+.8533653s^2}$$

$$P_{15} = \frac{.7783129+.2119030s+.1121355s^2}{-.7053800+.3652522s+.1207739s^2} \qquad P_{16} = \frac{.6856896+.1531217s+.6970851s^2}{-.8874894+.3835070s+.8999381s^2}$$

$$P_{17} = \frac{.8415518+.4062025s+.4094825s^2}{-.1858218+.1169181s+.4384902s^2} \qquad P_{18} = \frac{.8784126+.1138360s+.1998338s^2}{-.0922525+.8911246s+.8976057s^2}$$

$$P_{19} = \frac{.5618661+.5896177s+.6853980s^2}{-.3779664+.0634272s+.4911111s^2} \qquad P_{20} = \frac{.8906225+.5042213s+.3493615s^2}{-.9369316+.6263942s+.3402217s^2}$$

$$P_{21} = \frac{.3873779+.9222899s+.9488184s^2}{-.1980522+.6807702s+.9966154s^2} \qquad P_{22} = \frac{.3435337+.3760119s+.7340941s^2}{-.4702029+.2588597s+.0215396s^2}$$

$$P_{23} = \frac{.2615761+.4993494s+.2638578s^2}{-.3778819+.4394804s+.6876891s^2} \qquad P_{24} = \frac{.5253563+.5376230s+.1199926s^2}{-.3629649+.8598819s+.0044924s^2}$$

$$P_{25} = \frac{.2256303+.6274093s+.7608433s^2}{-.2793334+.5482823s+.6397855s^2} \qquad P_{26} = \frac{.0485566+.6723950s+.2017173s^2}{-.6721443+.8135995s+.5416148s^2}$$

$$P_{27} = \frac{.3911574+.8300317s+.5878720s^2}{-.6908951+.4898237s+.2247649s^2} \qquad P_{28} = \frac{.4829179+.2232865s+.8400886s^2}{-.5104758+.0240026s+.7364005s^2}$$

$$P_{29} = \frac{.1205996+.2855364s+.8607515s^2}{-.0610603+.7369196s+.4496365s^2} \qquad P_{30} = \frac{.8494102+.5257061s+.9931210s^2}{-.9335384+.2411156s+.9678138s^2}$$

$$P_{31} = \frac{.6488563+.9923191s+.0500420s^2}{-.1448998+.1529693s+.6489060s^2} \qquad P_{32} = \frac{.7485507+.4104059s+.6084526s^2}{-.5721211+.2648359s+.2544667s^2}$$

$$P_{33} = \frac{.8544211+.0642647s+.8279083s^2}{-.5459887+.4298193s+.1923975s^2} \qquad P_{34} = \frac{.9262344+.5667211s+.5711639s^2}{-.0516680+.7673939s+.6861492s^2}$$

$$P_{35} = \frac{.8160110+.0568928s+.5595937s^2}{-.5011013+.8753260s+.4231456s^2} \qquad P_{36} = \frac{.1249340+.7279222s+.2677766s^2}{-.9260972+.3796988s+.6734442s^2}$$

$$P_{37} = \frac{.5465335+.9885408s+.7395657s^2}{-.8546948+.3062357s+.8270697s^2} \qquad P_{38} = \frac{.0037173+.5900573s+.3096467s^2}{-.5857100+.3880052s+.9360806s^2}$$

$$P_{39} = \frac{.2552206+.6251879s+.1157417s^2}{-.8791871+.1047293s+.8326222s^2} \qquad P_{40} = \frac{.6117004+.6783956s+.3320095s^2}{-.2532048+.7603832s+.7424780s^2}$$

$$P_{41} = \frac{.0258710+.5174468s+.3916873s^2}{-.7903951+.0340930s+.0194640s^2} \qquad P_{42} = \frac{.2413538+.5064435s+.4236102s^2}{-.6329806+.1423966s+.8016032s^2}$$

$$P_{43} = \frac{.2893728+.0887932s+.6212882s^2}{-.8602324+.5554559s+.2554539s^2} \qquad P_{44} = \frac{.3454984+.7064868s+.5211472s^2}{-.9019620+.8031897s+.2688110s^2}$$

$$P_{45} = \frac{.2870401+.6502795s+.0881335s^2}{-.3158749+.5183992s+.7558491s^2} \qquad P_{46} = \frac{.4498763+.7227253s+.8976796s^2}{-.4001024+.1518561s+.4347590s^2}$$

$$P_{47} = \frac{.2427822+.4337721s+.9677053s^2}{-.1110896+.5884731s+.6425967s^2} \qquad P_{48} = \frac{.5068534+.5232976s+.5596948s^2}{-.4705128+.2545093s+.6957868s^2}$$

$$P_{49} = \frac{.5617307+.468176s+.7794547s^2}{-.5362763+.9992680s+.6499576s^2} \qquad P_{50} = \frac{.7901072+.9808542s+.8187066s^2}{-.2878432+.6398977s+.2303190s^2}$$

$$P_{51} = \frac{.4256872+.2461561s+.9229532s^2}{-.8740126+.5506716s+.5504368s^2} \qquad P_{52} = \frac{.1000746+.4678218s+.3950498s^2}{-.0740834+.4607008s+.3038997s^2}$$

$$P_{53} = \frac{.0366117+.5175369s+.8325452s^2}{-.2814957+.5933008s+.0371031s^2} \qquad P_{54} = \frac{.6104832+.1871112s+.0189575s^2}{-.7083768+.6538194s+.1703881s^2}$$

$$P_{55} = \frac{.8433565+.0748595s+.8532815s^2}{-.1652647+.4168341s+.5698686s^2} \qquad P_{56} = \frac{.0124590+.1867539s+.4920584s^2}{-.2667353+.9910155s+.0036411s^2}$$

$$P_{57} = \frac{.7489608+.9414957s+.2124056s^2}{-.8819408+.3720780s+.4491547s^2} \qquad P_{58} = \frac{.579502+.2628148s+.4360987s^2}{-.3537155+.7061824s+.0576811s^2}$$

$$P_{59} = \frac{.9110545+.8082667s+.8102653s^2}{-.0264621+.5780865s+.4956056s^2} \qquad P_{60} = \frac{.2590428+.4139087s+.3599928s^2}{-.0861647+.6023196s+.1888933s^2}$$

$$P_{61} = \frac{.6912788+.7656859s+.3572650s^2}{-.2313965+.5715097s+.5641231s^2} \quad P_{62} = \frac{.7693400+.5477634s+.0962289s^2}{-.1340646+.0549629s+.8562210s^2}$$

$$P_{63} = \frac{.9561172+.2207409s+.0143259s^2}{-.6724003+.1205854s+.0794764s^2} \quad P_{64} = \frac{.8191490+.1304993s+.9682004s^2}{-.1902998+.0143620s+.6196199s^2}$$

$$P_{65} = \frac{.6561381+.2445539s+.5283124s^2}{-.7307509+.0257951s+.224104s^2} \quad P_{66} = \frac{.8468926+.7876622s+.1262083s^2}{-.5048490+.8411249s+.8582095s^2}$$

$$P_{67} = \frac{.7883861+.3453042s+.2659857s^2}{-.8791159+.4381883s+.7114552s^2} \quad P_{68} = \frac{.9709819+.8875248s+.2066753s^2}{-.5544437+.4643402s+.8026028s^2}$$

$$P_{69} = \frac{.8525161+.6744698s+.9152874s^2}{-.2063297+.4197426s+.6153324s^2} \quad P_{70} = \frac{.0284860+.2367841s+.7015344s^2}{-.4011611+.8023654s+.8958183s^2}$$

$$P_{71} = \frac{.1202527+.8287412s+.3161073s^2}{-.4559329+.5228588s+.8721761s^2} \quad P_{72} = \frac{.5305191+.5715175s+.0478015s^2}{-.6977152+.5095121s+.8389768s^2}$$

$$P_{73} = \frac{.8248620+.5798843s+.2791808s^2}{-.3965912+.4531980s+.2040955s^2} \quad P_{74} = \frac{.9545111+.9071155s+.3360149s^2}{-.4983811+.7817818s+.3166538s^2}$$

$$P_{75} = \frac{.1175613+.9253724s+.7263671s^2}{-.1452423+.7765570s+.7006826s^2} \quad P_{76} = \frac{.9009498+.3948993s+.5655180s^2}{-.9075359+.0466059s+.2073105s^2}$$

$$P_{77} = \frac{.7061490+.6787831s+.4132936s^2}{-.1009295+.5113327s+.5222551s^2} \quad P_{78} = \frac{.1402291+.4952356s+.4194877s^2}{-.0177190+.7502102s+.9416421s^2}$$

$$P_{79} = \frac{.8626222+.2857510s+.2512136s^2}{-.7937671+.6883629s+.5211603s^2} \quad P_{80} = \frac{.3389102+.3921976s+.4681552s^2}{-.4309147+.0167756s+.4509842s^2}$$

$$P_{81} = \frac{.3361603+.5336877s+.2039064s^2}{-.5843379+.9345411s+.3961293s^2} \quad P_{82} = \frac{.1589990+.0181815s+.4098371s^2}{-.7954285+.5609864s+.7240870s^2}$$

$$P_{83} = \frac{.0105835+.1965310s+.2725595s^2}{-.1390616+.2205847s+.6724056s^2} \quad P_{84} = \frac{.3437655+.2033702s+.3011945s^2}{-.1012785+.7648541s+.2386146s^2}$$

$$P_{85} = \frac{.2762596+.2944531s+.5718074s^2}{-.3316809+.4816977s+.7124842s^2} \quad P_{86} = \frac{.2141770+.6895462s+.5855573s^2}{-.8344425+.1558786s+.3286494s^2}$$

$$P_{87} = \frac{.4204123+.4277572s+.3184586s^2}{-.0549941+.2476022s+.4837769s^2} \quad P_{88} = \frac{.5761894+.4254902s+.9761982s^2}{-.3696003+.5216529s+.7662767s^2}$$

$$P_{89} = \frac{.251896+.4391129s+.0759304s^2}{-.7453266+.0805230s+.3153839s^2} \quad P_{90} = \frac{.2559380+.0670617s+.7651132s^2}{-.0571812+.4409417s+.3489988s^2}$$

$$P_{91} = \frac{.0417362+.3438272s+.1970167s^2}{-.5999652+.8911934s+.7413818s^2} \quad P_{92} = \frac{.2122899+.3140399s+.7821625s^2}{-.2085742+.9780591s+.9702722s^2}$$

$$P_{93} = \frac{.0540932+.9190207s+.4603516s^2}{-.8798878+.4588094s+.1276511s^2} \quad P_{94} = \frac{.2992685+.0029166s+.8993471s^2}{-.4782810+.551044s+.3828862s^2}$$

$$P_{95} = \frac{.8387927+.4343749s+.7767876s^2}{-.7116099+.0960961s+.4882477s^2} \quad P_{96} = \frac{..1395318+.1150637s+.535542s^2}{-.8585865+.4196785s+.5191950s^2}$$

$$P_{97} = \frac{.4311733+.6145385s+.9258962s^2}{-.8078222+.7502557s+.0906856s^2} \quad P_{98} = \frac{.0993817+.4280579s+.9431831s^2}{-.5159963+.7218150s+.6833898s^2}$$

$$P_{99} = \frac{.0327395+.9213267s+.9449024s^2}{-.8917340+.2895541s+.5022272s^2} \quad P_{100} = \frac{.9007070+.8094316s+.0251954s^2}{-.8842349+.3907764s+.0540434s^2}$$

El controlador encontrado es:

$$C(s) = \frac{1.1162009 + 6.7436981s + 2.7276503s^2 + 3.7091082s^3}{0.0065512 + .7649549s + .0771613s^2 + .6299392s^3}$$

el la figura 3-2 se muestra la gráfica de la evolución de la mejor aptitud para este ejemplo, el controlador fue encontrado en 30 iteraciones.

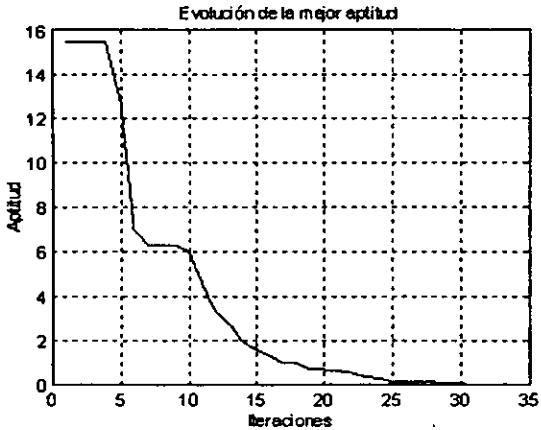

Figura 3-2: Evolucion de la mejor aptitud para el ejemplo 3.5

3.5 Conclusiones

Hemos planteado el problema de estabilización fuerte simultanea, como un problema de interés en el área de control robusto. Hemos dicho además que se trata de un problema complicado, para el que existen pocos métodos de solución y que estos métodos son muy restrictivos en el tipo de conjuntos de plantas en los que se aplica. Aqui se ha presentado un algoritmo usando estratégias evolutivas, se ha diseñado una función objetivo para el problema, así como las funciones de codificación y decodificación. Se le ha probado con ejemplos donde las plantas son de fase no mínima, conjuntos numerosos de hasta 100 plantas y con cambio de signo de alta frecuencia. El diseño sencillo con que se ha planteado la función objetivo ilustra la flexibilidad de su planteamiento y los resultados justifican ampliamente su aplicación en el diseño de controladores para este problema.

Capítulo 4

Observación simultánea

4.1 Introducción

Un problema que ha sido planteado recientemente pero que no ha sido estudiado con detalle, es el de observabilidad simultánea. Por ejemplo, Yao et al. [34] estudian el problema de diseñar un observador para n plantas LTI y SISO, usando técnicas de factorización coprima. En ese trabajo establecen condiciones necesarias y suficientes para la existencia de un observador simultáneo de n plantas. Chen et al. [7] replantean el problema de diseño de controladores para estabilización simultánea en términos de una estructura de observador generalizado, para sistemas MIMO. Sin embargo en ambos casos sus ejemplos solo incluyen dos sistemas a ser observados. Basados en el hecho de que el problema de diseñar observadores es al menos tan difícil como diseñar controladores. Es muy probable que al igual que el problema de estabilización simultánea este problema sea del tipo NP-hard.

4.2 Planteamiento del problema

Sea el conjunto de sistemas lineales e invariantes con el tiempo descritos por:

$$\begin{aligned} \dot{x}(t) &= A_i x(t) + B_i u(t) \\ y(t) &= C_i x(t) + D_i u(t) \end{aligned} \qquad i = 0, 1, ..., n$$

$x(t) \in R^n$ es el vector de estados, $u(t) \in R^r$ es el vector de entrada, y $y(t) \in R^m$ es el vector de

salida, con A_i, B_i, C_i, y D_i matrices constantes de dimensiones $n \times n$, $n \times j$, $m \times n$, y $m \times j$ respectivamente. Usando la transformación de Laplace a las ecuaciones previas se tiene que:

$$y(s) = G_i(s)u(s)$$

donde:

$$G_i(s) = C_i(sI - A_i)^{-1}B_i + D_i.$$

Se desea encontrar un observador que realice una estimación para:

$$z(t) = E_i x(t), \quad E_i \in R^{k \times n}.$$

Un observador para el sistema $G_i(s)$ es un sistema dinámico

$$r(s) = F(s)u(s) + H(s)y(s)$$

con la propiedad de que el error de estimación satisface:

$$\lim_{t \to \infty} (E_i x(t) - r(t)) = 0$$

para todo $u(s)$, donde $F(s)$ y $H(s)$ son matrices de funciones propias y estables. Si tal observador existe, decimos que $G_1(s)$, $G_2(s)$, ..., $G_n(s)$ son simultáneamente observadas.

El problema de observación simultánea es el siguiente: dadas n plantas $G_i(s)$, $i = 0, 1, ..., n$, queremos saber cuando es posible encontrar un observador común al conjunto de plantas dado.

4.3 Preliminares

Sea RH^∞ el dominio euclidiano de funciones reales, propias y estables.

y $R(s)$ el campo de las funciones racionales.

La factorización coprima por la derecha y factorización coprima por la izquierda de la planta $G_i(s)$ son

$$G_i(s) = N_i(s)M_i^{-1}(s) = \widehat{M}_i^{-1}(s)\widehat{N}_i(s)$$

respectivamente, y $N_i(s)$, $M_i(s)$ son matrices RH^∞ coprimas por la derecha, $\widehat{N}_i(s)$ y $\widehat{M}_i(s)$ son matrices RH^∞ coprimas por la izquierda.

Una doble factorización coprima de $G_i(s)$ esta formada de las dos factorizaciones coprimas anteriores y de matrices $Y_i(s)$, $X_i(s)$, $\widehat{Y}_i(s)$ y $\widehat{X}_i(s)$ en RH^∞ tales que

$$\begin{pmatrix} Y_i(s) & X_i(s) \\ -\widehat{N}_i(s) & \widehat{M}_i(s) \end{pmatrix} \begin{pmatrix} M_i(s) & -\widehat{X}_i(s) \\ N_i(s) & \widehat{Y}_i(s) \end{pmatrix} =$$

$$\begin{pmatrix} M_i(s) & \widehat{X}_i(s) \\ -N_i(s) & \widehat{Y}_i(s) \end{pmatrix} \begin{pmatrix} Y_i(s) & -X_i(s) \\ \widehat{N}_i(s) & \widehat{M}_i(s) \end{pmatrix} =$$

$$\begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix}$$

En el espacio de estados las matrices $M_i(s)$ y $N_i(s)$ de la factorización coprima de $G_i(s)$ están dadas como :

$$\begin{aligned} N_i(s) &= C_{k_i}(sI - A_{k_i})^{-1}B_i + D \\ M_i(s) &= K_i(sI - A_{k_i})^{-1}B_i + I \end{aligned}$$

donde K_i es una matriz tal que $A_{k_i} = A_i + B_iK_i$ es estable y $C_{k_i} = C_i + D_iK_i$.

Usando estados parciales podemos expresar a $z(s)$ como

$$z(s) = P_i(s)\xi(s)$$

donde $\xi(s)$ es el estado parcial y

$$P_i(s) = E_i(sI - A_{k_i})^{-1}B_i$$

ver [34]. Ahora el problema de diseñar un observador común para los sistemas $G_i(s)$, se puede

enunciar así : encontrar matrices en RH^∞, $F(s)$ y $H(s)$ tales que para toda $u(s)$

$$z(s) - r(s) = 0.$$

La última ecuación es equivalente a la siguiente identidad,

$$F(s)M_i(s) + H(s)N_i(s) = P_i(s).$$

La parametrzación del observador esta dada en el siguiente resultado.

Lema 4.1 [Ding] El conjunto de matrices en RH^∞, $F(s)$ y $H(s)$ que satisfacen

$$F(s)M(s) + H(s)N(s) = P(s)$$

es dada por

$$\begin{aligned} F(s) &= P(s)Y(s) - Q(s)\widehat{N}(s) \\ H(s) &= P(s)X(s) + Q(s)\widehat{M}_i(s), \end{aligned} \qquad Q(s) \in RH^\infty$$

Para cualquier matriz de dimensiones apropiadas $Q(s) \in RH^\infty$, $F(s)$ y $H(s)$ que satisfacen las dos ecuaciones anteriores, también satisface la primera ecuación para $P(s)$.

El siguiente resultado en [34] establece condiciones necesarias y suficientes para que tenga solución el problema de obsevación simultánea.. Ellos muestran que el problema de observación simultánea de $n+1$ plantas es equivalente al de observar simultáneamente n plantas con un observador común.

Lema 4.2 Dadas las plantas $G_i(s)$ para $i = 0, 1, ..., n$. Definamos

$$\begin{aligned} A_i(s) &= Y_0(s)M_i(s) + X_0(s)N_i(s) \\ B_i(s) &= -\widehat{N}_0(s)M_i(s) + \widehat{M}_0(s)N_i(s)\,. \end{aligned}$$

Entonces las plantas $G_i(s)$ se pueden observar simultáneamente si y sólo si existe una matriz $R(s)$ en RH^∞, tal que

$$P_0(s)A_i(s) + R(s)B_i(s) = P_i(s)$$

para $i = 1, ..., n$.

La formula explícita que dan para el observador en el caso de dos plantas es :

$$\begin{aligned} r(s) &= \left[P_0(s)Y_0(s) - R(s)\widehat{N}_0(s)\right] u(s) \\ &\quad + \left[P_0(s)X_0(s) + R(s)\widehat{M}_0(s)\right] y(s) \end{aligned}$$

Una observación fundamental que no hicieron Yao et al, es que este observador también puede observar simultáneamente n plantas, porque éste también puede observar simultáneamente a las plantas $G_0(s)$ y $G_i(s)$ para $i = 1, 2, ...n$, por su teorema 1 aplicado a las plantas $G_0(s)$ y $G_i(s)$ con i fijo, y como la parametrización del observador sólo depende de la doble factorización coprima de la planta $G_0(s)$. Entonces este observador puede observar simultáneamente a las plantas $G_i(s)$ para $i = 1, 2, ..., n$ y a la planta auxiliar $G_0(s)$. Por lo tanto, el problema se reduce a encontrar una $R(s)$ tal que cumpla con la ecuación de la $P_i(s)$ del lema 4.2.

4.4 Aproximación usando estrategias evolutivas

4.4.1 Codificación y decodificación

Para resolver el problema de encontrar un observador que observe a un conjunto finito de plantas, podemos hacer referencia al lema 4.2, que establece condiciones necesarias y suficientes para la existencia de dicho observador. En forma simplificada el problema se reduce a encontrar una matriz RH^∞, que cumpla con las condiciones que establece el sistema de ecuaciones del lema 4.2.

La definición de las variables objeto en relación a este problema nos sugiere que al igual que en el problema de estabilización simultánea, se pueden plantear varias opciones:

1. Las variables objeto son los coeficientes de las funciones racionales que son entradas de la matriz RH^∞ que se busca, o

2. Las variables objeto son los polos, ceros y ganancia de alta frecuencia de estas mismas funciones racionales.

Si se elige la primera, al igual que en el problema de estabilización fuerte simultánea, el dominio de cada una son los números reales. En cambio si se elige la segunda el dominio de las variables objeto serán los números complejos.

Eligiendo la primera de ellas se simplifica la función de codificación y decodificación del algoritmo. Cada elemento del individuo tiene correspondencia con un coeficiente de los polinomios numerador y denominador de las funciones racionales que son entradas de la matriz $R(s)$ del lema 4.2.

Una vez que hemos definido el dominio de las variables objeto, hay que definir las funciones de codificación y decodificación a un vector en $\Re^n$ con el que se trabaja en las estrategias evolutivas. La función de decodificación se expresa como:

Definición 4.1 Sea $x \in \Re^n$ la transformación vector a matriz de funciones racionales se define como: $r : \Re^n \rightarrow \Re^{m \times u}(s)$. Tal que $r(x, m, u) = [y_{ij}]$ donde:

$$y_{ij} = \frac{pol\left(\left[{}^{1}_{(i-1)u+jk-k+1} sub^{1}_{((i-1)u+j)k-\frac{k}{2}}(x), s\right]\right)}{pol\left(\left[{}^{1}_{((i-1)u+j)k-\frac{k}{2}+1} sub^{1}_{((i-1)u+j)k}(x), s\right]\right)}$$

y $k = \frac{n}{um}$.

4.4.2 Función objetivo

Dadas las matrices $A_i(s)$, $B_i(s)$ y $P_i(s)$ del lema 4.2, la función objetivo debe ser tal que encuentre una $R(s)$ que satisfaga las condiciones del mismo. Estas condiciones son dos:

1. Que se satisfaga el sistema de ecuaciones descrito como:
   $P_0(s) A_i(s) + R(s) B_i(s) = P_i(s)$
2. Que $R(s) \in \Re H^\infty$.

La primera condición puede replantearse como un error que hay que minimizar. Este error es: $e(s) = P_0(s) A_i(s) + R(s) B_i(s) - P_i(s)$ y buscamos que $e(s) \approx 0$. La segunda es un requisito que limita el dominio de los individuos a funciones racionales estables, la más simple de las aproximaciones resuelve este problema penalizando al individuo cuando no cumple con esta condición.

Así nuestra función objetivo queda dada por:

$$f(x) = \beta \sum_{i=1}^{N} \left\| {}_{1}^{1}sub_{\frac{1}{2}}^{1} \left(r^{-1}(P_0(s) \cdot A_i(s) + r(x,m,u) \cdot B_i(s) - P_i(s)) \right) \right\|$$

donde:

$$\beta = K \sum_{k=1}^{m} \sum_{l=1}^{u} \gamma(r(x,m,u)_{kl}) + 1$$

$$\gamma(x) = \begin{cases} 1 & \text{si} \quad x \notin \Re H^{\infty} \\ 0 & \text{si} \quad x \in \Re H^{\infty} \end{cases}$$

Obsérvese que en la función objetivo solo importa minimizar el error para los coeficientes de los polinomios de los numeradores. Esto debe ser así, puesto que el minimizar los coeficientes de ambos polinomios en las funciones racionales no es en general conveniente ya que el error real puede mantenerse constante a pesar de que los coeficientes desciendan si el descenso es proporcional en ambos polinomios. En cambio si se minimiza únicamente el error producido por los numeradores se tendrá que los elementos de la matriz $R(s)$ tienden a cero a pesar de que los denominadores no.

En este trabajo se ha elegido la constante de penalización $K = 1000$ con la que se obtuvieron buenos resultados.

4.5 Ejemplos

En esta sección se presentan ejemplos de observadores para varios tipos de plantas, la función objetivo es la descrita en la sección anterior. Para todos los ejemplos se han usado los mismos parámetros del algoritmo: $\mu = 7, \lambda = 49$.

Ejemplo 4.1 En este ejemplo se presentan 7 plantas de tercer orden y grado relativo uno. para

las que : $E_i = \begin{bmatrix} 0 & -2 & 4 \end{bmatrix} \qquad i = 0, ..., 6$

$$
\begin{array}{lll}
G_0 = \frac{24s^2+138s+69}{s^3+8s^2+13s+6} & P_0 = \frac{-6s^2+78s+264}{s^3+8s^2+13s+6} & A_i = 1 \qquad i = 0, ..., 6 \\
G_1 = \frac{24s^2+13s-56}{s^3+3s^2+3s+1} & P_1 = \frac{-6s^2+128s-314}{s^3+3s^2+3s+1} & B_1 = \frac{-5s^2-185s-405}{s^4+9s^3+21s^2+19s+6} \\
G_2 = \frac{24s^2+38s-31}{s^3+4s^2+5s+2} & P_2 = \frac{-6s^2+118s+304}{s^3+4s^2+5s+2} & B_2 = \frac{-4s^2-148s-324}{s^4+10s^3+29s^2+32s+12} \\
G_3 = \frac{24s^2+93s+24}{s^3+6s^2+9s+4} & P_3 = \frac{-6s^2+78s+264}{s^3+6s^2+9s+4} & B_3 = \frac{3s^2-39s-132}{s^4+12s^3+45s^2+58s+24} \\
G_4 = \frac{24s^2+83s+14}{s^3+5s^2+7s+3} & P_4 = \frac{-6s^2+28s+214}{s^3+5s^2+7s+3} & B_4 = \frac{17s^2+29s-123}{s^4+11s^3+37s^2+45s+18} \\
G_5 = \frac{24s^2+68s+39}{s^3+6s^2+9s+4} & P_5 = \frac{-6s^2+178s+204}{s^3+6s^2+9s+4} & B_5 = \frac{-22s^2-174s-42}{s^4+12s^3+45s^2+58s+24} \\
G_6 = \frac{24s^2+213s+179}{s^3+10s^2+17s+8} & P_6 = \frac{-6s^2-42s+4}{s^3+10s^2+17s+8} & B_6 = \frac{27s^2+284s-522}{s^4+16s^3+77s^2+110s+48}
\end{array}
$$

los elementos de la factorización coprima son:

$Y_i = \widehat{Y}_i = M_i = \widehat{M}_i = 1$

$X_i = \widehat{X}_i = 0$

$N_i = \widehat{N}_i = G_i$

se tiene que: $R = -4$.

El observador encontrado es:

$$r(s) = \frac{90}{s+1} u(s) - 4y(s) .$$

Este es el primer ejemplo de mas de dos plantas reportado en la literatura. Además obsérvese que no todas las plantas son de fase mínima. Para este caso la mejor aptitud encontrada es de 9.93×10^{-9} tras 501 iteraciones.

Ejemplo 4.2 En este ejemplo se tienen 8 plantas de segundo grado con el mismo grado relativo

, $E_i = \begin{bmatrix} 1 & -3 \end{bmatrix} \qquad i = 0, ..., 7$

$$
\begin{array}{llll}
G_0 = \frac{10s}{s^2+4s+3} & P_0 = \frac{-5s+3}{s^2+4s+3} & A_i = 1 & i = 0, ..., 7 \\
G_1 = \frac{10s}{s^2+6s+5} & P_1 = \frac{-5s+5}{s^2+6s+5} & B_1 = \frac{-20s}{s^3+9s^2+23s+15} & \\
G_2 = \frac{10s-20}{s^2+2s+1} & P_2 = \frac{-5s+13}{s^2+2s+1} & B_2 = \frac{-60}{s^3+5s^2+7s+3} & \\
G_3 = \frac{10s-20}{s^2+3s+2} & P_3 = \frac{-5s+14}{s^2+3s+2} & B_3 = \frac{-10s-60}{s^3+6s^2+11s+6} & \\
G_4 = \frac{10s}{s^2+5s+4} & P_4 = \frac{-5s+4}{s^2+5s+4} & B_4 = \frac{-10s}{s^3+8s^2+19s+12} & \\
G_5 = \frac{10s}{s^2+8s+7} & P_5 = \frac{-5s+7}{s^2+8s+7} & B_5 = \frac{-40}{s^3+11s^2+31s+21} & \\
G_6 = \frac{10s-20}{s^2+9s+8} & P_6 = \frac{-5s+20}{s^2+9s+8} & B_6 = \frac{-70s-60}{s^3+12s^2+35s+24} & \\
G_7 = \frac{10s-20}{s^2+7s+6} & P_7 = \frac{-5s+18}{s^2+7s+6} & B_7 = \frac{-50s-60}{s^3+10s^2+27s+18} &
\end{array}
$$

con los siguientes elementos de la factorización coprima:

$Y_i = \widehat{Y}_i = M_i = \widehat{M}_i = 1$

$X_i = \widehat{X}_i = 0$

$N_i = \widehat{N}_i = G_i$

donde : $R = -0.6$.

El observador queda como:

$$r(s) = \frac{1}{s+1}u(s) - 0.6y(s) .$$

En este ejemplo algunas de las plantas son de fase no mínima. La mejor aptitud fue de 6.765×10^{-12} tras 270 iteraciones.

Ejemplo 4.3 Para este ejemplo se tienen plantas de diferente orden y mismo grado relativo, con $E_i = \begin{bmatrix} 0 & -2 & 1 \end{bmatrix} \qquad i = 0, 1, 2$

$$
\begin{array}{lll}
G_0 = \frac{-6s^2-8s-5}{s^3+2.25s^2+1.5s+.25} & P_0 = \frac{-5s+1}{s^2+1.25s+.25} & A_i = 1 \qquad i = 0, 1, 2. \\
G_1 = \frac{-6}{s+1} & P_1 = \frac{-5s-3}{s^2+2s+1} & B_1 = \frac{-.5s+3.5}{s^3+2.25s^2+1.5s+.25} \\
G_2 = \frac{-6s^2-9.1428571s-5.1428571}{s^3+2.25s^2+1.5s+.25} & P_2 = \frac{-5s-.4285714}{s^2+1.5s+.5} & B_2 = \frac{.3571429s+1.2142857}{s^3+1.75s^2+.875s+.125}
\end{array}
$$

los elementos de la factorización coprima son:

$Y_i = \widehat{Y}_i = M_i = \widehat{M}_i = 1$

$X_i = \widehat{X}_i = 0$

$N_i = \widehat{N_i} = G_i$

Se encontró $R = -1/2$.

El observador es:

$$r(s) = \frac{-8s - 6}{s^2 + 2s + 1} u(s) - \frac{1}{2} y(s) .$$

En este caso la dificultad que presentan estas plantas es que son de orden diferente. La codificación demanda que se trate de plantas en el espacio de estados. Es interesante notar que en la planta G_1 hubo una doble cancelación de polos y ceros. En este ejemplo la mejor aptitud es: 1.757×10^{-12} tras 283 iteraciones.

4.6 Conclusiones

Se ha presentado en este capítulo un problema de control robusto conocido como problema de observación simultánea. Se presentó una función de codificación y otra de decodificación para que el problema pudiera ser resuelto con estrategias evolutivas. Además se ha definido una función objetivo posible para la solución del mismo, encontrándose los primeros observadores para conjuntos de más de dos plantas reportados en la literatura. Los resultados sugieren la factibilidad de su utilización al abordar este problema, pero también mustran la dificultad del mismo, puesto que la convergencia a la solución se da en un número mayor de iteraciones comparado con estabilización simultánea.

Capítulo 5

Conclusiones

Se ha presentado un algoritmo de optimización aplicado a dos problemas interesantes de control que no tienen solución analítica total: el problema de estabilización fuerte simultánea y el problema de observación simultánea. Ambos problemas no están resueltos en su totalidad y en el caso del problema de observación simultánea no existen resultados teóricos para más de dos plantas.

En esta tesis hemos presentado ejemplos de estos problemas cuya solución fue encontrada usando estrategias evolutivas. Se han elegido problemas interesantes por el número de plantas involucradas, porque se trabaja con plantas de fase no mínima, con plantas de grado relativo diferente y con cambios de signo de alta frecuencia en las plantas.

Subrayando que el problema de estabilización fuerte simultánea es de tipo NP-Duro y el problema de observación simultánea quizá también lo sea, la observación directa de los resultados nos conduce a aceptar a las estrategias evolutivas como una herramienta útil aplicable a la búsqueda de controladores y observadores en problemas de control robusto.

Capítulo 6

Apéndice A

6.1 Programas en SCILAB

En esta sección se presenta la implementación de los algoritmos de estabilización fuerte simultánea y de observación simultánea. Los programas están desarrollados en SCILAB [32].

Se ha utilizado una versión de estrategias evolutivas desarrollada por Rodolfo A. Sánchez del Laboratorio de INvestigación para el Desarrollo Académico [20] que se implementó sobre esta plataforma.

6.1.1 Estabilización Fuerte Simultánea.

Para este problema se tiene la función objetivo en el archivo *f_stab_strong.sci*, la función es *asig_apt*, esta es debe ser invocada por el programa *evol_str*, que tiene como parámetros el tipo de estrategia, el número de padres, hijos y otros parámetros de control.

archivo:f_stab_strong.sci

```
function [fitness_values]=asig_apt(population)
    // funcion para encontrar un controlador que estabiliza simultaneamente
    // el conjunto de plantas Pi
    // %Pi es una lista con las plantas representadas como funciones de transferencia
    epsilon=.1;
    N=size(%Pi);
```

```
[m,n]=size(population);
for g=1:m
  Cnum=poly(population(g,1:n/2),'s','coeff');
  Cden=poly(population(g,n/2+1:n),'s','coeff');
  fitness_values(g)=0;
  for i=1:N
    Hi=numer(%Pi(i))*Cnum+denom(%Pi(i))*Cden;
    R=real(roots(Hi));
    indices=find(R>=0)
    if(size(indices,1)==0)
      ;
    else
      fitness_values(g)=fitness_values(g)+sum(tanh(R(indices)+epsilon));
    end
  end
  pena=size(find(real(roots(Cden))>=0));
  if(pena(1)~=0)
    fitness_values(g)=fitness_values(g)*1.e9;
  end
end
```

6.1.2 Observación simultánea

El programa principal para el problema de observación simultánea es obsim.sci, este archivo contiene una función llamada *simul_observ* que tiene como parámetros de entrada una lista con los sistemas en Systems y las restricciones (E_i). La llamada a las estrategias evolutivas se hace con la función *evol_str*, nótese que la factorización coprima también se resuelve utilizando estrategias evolutivas. La función objetivo para el problema de observación simultánea se encuentra en el archivo *mateq.abs.sci*, bajo el nombre de asig_apt.

Archivo: obsim.sci

```
function [F,H,Pi,Asi,Bsi,R,Npi,Mpi,Ni,Mi,Xi,Yi,Xpi,Ypi,Ki,Fi,History]=...
    simul_observ(Systems,Ei)
    // funcion que encuentra el observador comun para las plantas descritas por
    // los sistemas en Systems y dada la lista de restricciones Ei.
    // El observador esta expresado como:
    // r(s)=F(s)u(s)+H(s)y(s)
    // con la propiedad de que el error de estimacion satisface:
    // lim ( Ei*x(t) - r(t) ) = 0
    // t->inf
    // Systems es una lista donde cada elemento es un sistema descrita por syslin
    lines(%inf);   // ajusta el despliegue para que no hayan interrupciones en el mismo
    n=size(Systems);  // n es el numero de systemas
    Ai=list();Bi=Ai;Ci=Ai;Di=Ai;Pi=Ai;Asi=Ai;Bsi=Ai;  // Se crean listas.
    for i=1:n,
     Ai(i)=Systems(i)(2);
     Bi(i)=Systems(i)(3);
     Ci(i)=Systems(i)(4);
     Di(i)=Systems(i)(5);
     temp=(obsvss(Systems(i)));
     Ob(i,1:2)=(size(temp(2))-size(Systems(i)(2)));
    end
    if(find(Ob~=0)~=[])    //Verifica si los sistemas son observables
     disp('Por lo menos uno de los sistemas no es observable')
     if(input('Deseas continuar s/n?',"s")=='n')
      return;
     end
    end
    // Aqui se calcula factorización coprima
    [Npi,Mpi,Ni,Mi,Xi,Yi,Xpi,Ypi,Ki,Fi]=coprime(Ai,Bi,Ci,Di);
```

```
disp(Fi,'Fi:',Ki,'Ki:',Ypi,'Ypi:',Xpi,'Xpi:',Yi,'Yi:');
disp(Xi,'Xi:',Mi,'Mi:',Ni,'Ni:',Mpi,'Mpi:',Npi,'Npi:');
for i=1:n,
 Pi(i)=clean(Ei(i)*inv(%s*eye-(Ai(i)+Bi(i)*Ki(i)))*Bi(i));
 Asi(i)=clean(Yi(1)*Mi(i)+Xi(1)*Ni(i));
 Bsi(i)=clean(-Npi(1)*Mi(i)+Mpi(1)*Ni(i));
end
disp(Pi,'Pi(s):');disp(Asi,'Ai(s)');disp(Bsi,'Bi(s)');
[m,n]=size(Pi(1));
[u,v]=size(Bsi(1));
if(%set(1)=='s') //Grado predeterminado
 M=(%set(2)+1)*2;
else
 M=(max(max(degree(numer(Pi(1)))),max(degree(denom(Pi(1))))+1)*2;
end
disp(M/2-1,'Order(R):');
M=M*m*u;
k=M/(m*u);
%Pi=Pi;%Asi=Asi;%Bsi=Bsi;
getf([%path+%set(6)],'c');    //compila la funcion objetivo
[solution,History]=evol_str(%set(3),%set(4),M,7*%set(4),1.e-6,%set(5),...
%set(8),%set(9));
for i=1:m,
 for j=1:u,
  R(i,j)=poly(solution(((i-1)*u+j)*k-k+1:((i-1)*u+j)*k-k/2),'s','coeff')/...
  poly(solution(((i-1)*u+j)*k-k/2+1:((i-1)*u+j)*k),'s','coeff');
 end
end
disp(R,'R(s):');
F=Pi(1)*Yi(1)-R*Npi(1);
```

```
H=Pi(1)*Xi(1)+R*Mpi(1);
disp(F,'F(s)');
disp(H,'H(s)');
if(exists('%test'))
 disp('resume to abort','testing mode');
 pause;
end
```

archivo: factorizacion_coprima.sci

```
function [Npi,Mpi,Ni,Mi,Xi,Yi,Xpi,Ypi,Ki,Fi]=coprime(Ai,Bi,Ci,Di)
    // funcion para calcular la factorizacion coprima de un conjunto de N+1 plantas
    // definidas en el espacio de estados.
    // Ai, Bi, Ci y Di . son listas con las matrices A, B, C y D de cada uno de
    // los sistemas.
    // Npi,Dpi,Ni,Di,Xi,Yi,Xpi y Ypi : son listas con las matrices de la factorizacion:
    //
    //
    n=size(Ai);
    AFi=list();
    AKi=AFi;Npi=AFi;Mpi=AFi;Ni=AFi;Mi=AFi,Xi=AFi;Yi=AFi;
    Xpi=AFi;Ypi=AFi;Ki=AFi;Fi=AFi;
    for i=1:n
     Ki(i)=stabilizable(Ai(i),Bi(i));
     Fi(i)=detectable(Ai(i),Ci(i));
     AFi(i)=clean(Ai(i)+Fi(i)*Ci(i));
     AKi(i)=clean(Ai(i)+Bi(i)*Ki(i));
     SI_AF=inv(%s*eye-AFi(i));
     SI_AK=inv(%s*eye-AKi(i));
     BFD=(Bi(i)+Fi(i)*Di(i));
     CKi=Ci(i)+Di(i)*Ki(i);
```

```
Npi(i)=clean(Ci(i)*SI_AF*BFD+Di(i));
Mpi(i)=clean(Ci(i)*SI_AF*Fi(i)+eye);
Ni(i)=clean(CKi*SI_AK*Bi(i)+Di(i));
Mi(i)=clean(Ki(i)*SI_AK*Bi(i)+eye);
Xi(i)=clean(Ki(i)*SI_AF*Fi(i));
Yi(i)=clean(eye-Ki(i)*SI_AF*BFD);
Xpi(i)=clean(Ki(i)*SI_AK*Fi(i));
Ypi(i)=clean(eye-CKi*SI_AK*Fi(i));
end
//****
//****
function [K]=stabilizable(A,B)
// Determina si un sistema es estabilizable.
//
// K es tal que: A+B*K es Hurwitz
%A=A;
%B=B;
getf([%path+'apt1.sci'],'c');
if(and(clean(real(spec(%A)))<0))
  K=zeros(size(%B,2),size(%A,1));
else
 solution=evol_str(1,7,size(%B,2)*size(%A,1),50,0,'n',%set(8),...
 %set(9));
 disp('stabilizable condition');
 k=1;
 for i=1:size(%B,2),
  for j=1:size(%A,1),
   K(i,j)=solution(k);
   k=k+1;
  end,
```

```
 end,
end
if(and(real(spec(A+B*round(K)))<0))
 K=round(K);
end
//******
//******
function [F]=detectable(A,C)
// Determina si un sistema es detectable.
//
// F es tal que: A+F*C es Hurwitz
 %A=A;
 %C=C;
getf([%path+'apt2.sci'],'c');
if(and(clean(real(spec(%A)))<0))
 F=zeros(size(%A,1),size(%C,1));
else
 solution=evol_str(1,7,size(%A,1)*size(%C,1),50,0,'n',%set(8),...
 %set(9));
 disp('Detectable condition');
 k=1;
 for i=1:size(%A,1),
  for j=1:size(%C,1),
   F(i,j)=solution(k);
   k=k+1;
  end,
 end,
end
if(and(real(spec(A+round(F)*C))<0))
 F=round(F);
```

```
end
```

archivo: apt1.sci

```
function [fitness_values]=asig_apt(population)
    // Funcion objetivo para encontrar la matriz K tal que
    // Ak=A+B*K es estable.
    // Minimizacion (la matriz es estable si la aptitud es cero).
    epsilon=.01;
    mat=[];
    row=size(population,1);
    n=size(%A,1);
    m=size(%B,2);
    for i=1:row,
     for j=1:m,
      for k=1:n,
       mat(j,k)=population(i,(j-1)*n+k);
      end,
     end
     mat=%A+%B*mat;
     eigval=spec(mat);
     x=find(real(eigval)>=0);
     if x==[] then
      fitness_values(i,1)=0;
     else
      fitness_values(i,1)=sum(tanh(real(eigval(x))+epsilon))/...
      prod(size(eigval));
     end,
     mat=[];
    end
```

archivo: apt2.sci

```
function [fitness_values]=asig_apt(population)
    // Funcion objetivo para encontrar la matriz F tal que
    // Ak=A+F*C es estable.
    // Minimizacion (la matriz es estable si la aptitud es cero).
    epsilon=.01;
    mat=[];
    row=size(population,1);
    n=size(%A,1);
    m=size(%C,1);
    for i=1:row,
     for j=1:n,
      for k=1:m,
       mat(j,k)=population(i,(j-1)*m+k);
      end,
     end
     mat=%A+mat*%C;
     eigval=spec(mat);
     x=find(real(eigval)>=0);
     if x==[] then
      fitness_values(i,1)=0;
     else
      fitness_values(i,1)=sum(tanh(real(eigval(x))+epsilon))/...
      prod(size(eigval));
     end,
     mat=[];
    end
```

archivo: mateq.abs.sci

```
function [fitness_values]=asig_apt(population)
```

```
// funcion para verificar si R satisface la condicion siguiente:
//
// P (s)*A (s)+R(s)*B (s)=P (s)
// para todos los sistemas
//
N=size(%Pi);
[m,n]=size(%Pi(1));
[u,v]=size(%Bsi(1));
[L,M]=size(population);
k=M/(m*u);
 for g=1:L,
 for i=1:m,
  for j=1:u,
   R(i,j)=poly(population(g,((i-1)*u+j)*k-k+1:((i-1)*u+j)*k-k/2),'s','coeff')/...
   poly(population(g,((i-1)*u+j)*k-k/2+1:((i-1)*u+j)*k),'s','coeff');
  end
 end
 fitness_values(g)=0;
 if(exists('%print'))
  disp(R,'R:');
 end
 for i=2:N
  E=(%Pi(1)*%Asi(i)+R*%Bsi(i)-Pi(i));
  if(exists('%print'))
   disp(%Pi(1)*%Asi(i)+R*%Bsi(i),'P0(s)*Ai(s)+R(s)*Bi(s)');
   disp(Pi(i),'Pi(s)');
   disp(E,'Matriz de error');
   disp(clean(E,1.e-6,1.e-6),'Clean(Error,1.e-6):')
  end
  n=coeff(numer(E));
```

```
fitness_values(g)=fitness_values(g)+sum(abs(n));
end
denominadores=denom(R);
for i=1:m
 for j=1:u
  if(~(clean(real(roots(denominadores(i,j))))<0))
   fitness_values(g)=fitness_values(g)*1000;
  end
 end
end
end
```

6.2 Programas en MATLAB

En esta sección se presenta una implementación alternativa para el problema de estabilización fuerte simultánea sobre MATLAB. Se presenta el programa *strong.m*, que es una estrategia evolutiva simplificada, para la codificación de polos, ceros y ganancia de alta frecuencia (ver apéndice B). La función objetivo esta en el archivo $f\_ob.m$.

6.2.1 Estabilización Fuerte Simultánea.

Para el problema de estabilización fuerte se presenta una versión simplificada de la estrategia $(\mu + \lambda) - ES$, el programa es el siguiente:

archivo:strong.m

```
function [contr_num,contr_den,ganancia_planta]=strong(pn,pd,w)
    % Estrategia-Evolutiva.
    % estabilizacion fuerte simultanea
    % [contr_num,contr_den,ganancia_planta]=strong(pn,pd,w)
    %parámetros iniciales
    n_individuos = input('Enter number of parents ');
```

```
size_individuo = input('Enter # variables by parent ');
n_hijos = input('Enter number of sons ');
n_iteraciones = input('Enter number of iterations ');
% fin parámetros iniciales
% inicialización
factor1=max(max(abs(mulroots(pn))));
factor2=max(max(abs(mulroots(pd))));
factor=max([factor1 factor2]);
rand('seed',sum(100*clock))
rand('uniform')
ganancia=(rand(n_individuos,1)-.5)*(2*factor);
actual_pob = rand(2*n_individuos,floor(size_individuo/2))-.5+...
(rand(2*n_individuos,floor(size_individuo/2))-.5)*i;
 pasos = rand(2*n_individuos,floor(size_individuo/2))-.5+...
(rand(2*n_individuos,floor(size_individuo/2))-.5)*i;
if(rem(size_individuo,2)==1)
  actual_pob(:,floor(size_individuo/2)+1)=rand(2*n_individuos,1)-.5;
  pasos(:,floor(size_individuo/2)+1)=rand(2*n_individuos,1)-.5;
end
actual_pob=actual_pob*2*factor;
pasos=pasos*2*factor;
aptitudes = zeros(n_individuos,1);
new_hijo = zeros(1,floor(size_individuo/2)+rem(size_individuo,2));
for(var=1:n_individuos)
  aptitudes(var,1)=f_ob(pn,pd,ganancia(var),actual_pob(2*var-1,:),...
  actual_pob(2*var,:)); % se asigna_aptitud.
end; % var
[best_apt,best_ind]= max(aptitudes);
mean_apt=mean(aptitudes);
best_apt_ant=0;
```

```
contar=0;
orig_pob=actual_pob;
orig_apt=aptitudes;
% fin inicialización
% estadística
fprintf('Iteration 0 Best_quality %f Average_fitness %f \n', best_apt,...
mean_apt);
% fin estadística
% etapa de evolucion
for iteracion=1:n_iteraciones
 rand('seed',sum(100*clock))
 paso_norma=mean(pasos);
 ganancia_norma=mean(ganancia);
 new_pob=actual_pob;
 for hijos=1:n_hijos
  box=randn(floor(size_individuo/2)+rem(size_individuo,2),2)/...
  size_individuo;
  for(var=1:floor(size_individuo/2)+rem(size_individuo,2))
   if(rand<0.5)
    paso_hijo(1,var)=paso_norma(var)*1.5;
   else
    paso_hijo(1,var)=paso_norma(var)/1.5;
   end %if
   if(rand<0.5)
    paso_hijo(2,var)=paso_norma(var)*1.5;
   else
    paso_hijo(2,var)=paso_norma(var)/1.5;
   end % if
   if(var==floor(size_individuo/2)+rem(size_individuo,2))
    paso_hijo(:,var)=real(paso_hijo(:,var));
```

```
end
num_ale=(((n_individuos-1)*rand+1)*2)-1;
new_hijo(1,var)=actual_pob(num_ale,var)+(paso_hijo(1,var)*...
box(var,1));
new_hijo(2,var)=actual_pob(num_ale+1,var)+(paso_hijo(2,var)*...
box(var,2));
end % var
if rand <0.5
 ganancia_new_hijo=ganancia_norma*1.5;
else
 ganancia_new_hijo=ganancia_norma/1.5;
end % if
aptitud_h=f_ob(pn,pd,ganancia_new_hijo,new_hijo(1,:),...
new_hijo(2,:)); % se asigna_aptitud.
[peor_apt,peor_ind]=min(aptitudes);
if aptitud_h >= peor_apt
 aptitudes(peor_ind,1)=aptitud_h;
 ganancia(peor_ind,1)=ganancia_new_hijo;
 pasos(peor_ind*2-1,1:size(paso_hijo,2))=paso_hijo(1,:);
 pasos(peor_ind*2,1:size(paso_hijo,2))=paso_hijo(2,:);
 new_pob((peor_ind*2)-1:peor_ind*2,:)=new_hijo;
end; % if
end % hijos
mean_apt=mean(aptitudes);
[best_apt,best_ind]= max(aptitudes);
fprintf('Iteration %d Best_quality %f Average-fitness %f \n',iteracion,...
best_apt,mean_apt);
if(best_apt_ant==best_apt)
 contar=contar+1;
 else
```

```
 contar=0;
 best_apt_ant=best_apt;
end
if(contar==7)
 [new_pob,pasos,ganancia]=bomba(new_pob,pasos,ganancia);
 for(i=1:n_individuos)
  aptitudes(i,1)=f_ob(pn,pd,ganancia(i),new_pob(2*i-1,:),...
  new_pob(2*i,:)); % se asigna_aptitud.
 end %i
 [peor_apt,peor_ind]=min(aptitudes);
 aptitudes(peor_ind)=best_apt;
 ganancia(peor_ind)=ganancia(best_ind);
 pasos(peor_ind*2-1,1:size(pasos(best_ind,:),2))=pasos(best_ind*2-1,:);
 pasos(peor_ind*2,1:size(pasos(best_ind,:),2))=pasos(best_ind*2,:);
 new_pob(peor_ind*2-1,:)=actual_pob(best_ind*2-1,:);
 new_pob(peor_ind*2,:)=actual_pob(best_ind*2,:);
 fprintf('\nBomba\n');
 contar=0;
end
actual_pob=new_pob;
if(best_apt>=1)
 fprintf('\n< 1 >\n');
 statics(iteracion)=1;
 break;
end % if
statics(iteracion)=best_apt;
end % iteracion
% fin etapa de evolución
% asignación
contr_num=polimiza(cerrconj(actual_pob((best_ind*2)-1,:)));
```

```
    contr_den=polimiza(cerrconj(actual_pob(best_ind*2,:)));
    ganancia_planta=ganancia(best_ind);
    plot(statics);
    save(w);
    % fin asignación
```

archivo:f_ob.m

```
function fit=f_ob(Pn,Pd,ganancia,Cz,Cp)
    % fitness=f_objetiv(Pn,Pd,ganancia,Cz,Cp)
    Cz=cerrconj(Cz);
    Cp=cerrconj(Cp);
    fit=0;
    n=size(Pn,1);
    if(m_rang(Cz)>m_rang(Cp))
      return;
    end
    Cn=polimiza(Cz);
    Cd=polimiza(Cp);
    pena=roots(Cd);
    if(size(find(real(pena)>=0),1)~=0)
      factor=1;
    else
     factor=0;
    end
    if(findconj(Cz) & findconj(Cp))
     for(i=1:n)
      H=sumpoly(ganancia*conv(Cn,Pn(i,:)),conv(Cd,Pd(i,:)));
      h=roots(H)';
      fit=fit+subfit(h)/n;
     end
```

Capítulo 7

Apéndice B

7.1 Estrategia simplificada para estabilización fuerte simultánea

En esta estrategia simplificada se seleccionan n individuos (padres) de la población (en este caso se seleccionan todos), los cuales se combinan produciendo una población de hijos. Cada uno de los individuos de esta población de hijos, se muta, decodifica y se le asigna el valor de aptitud correspondiente. Se junta la población inicial y la de hijos, seleccionándose los individuos más aptos que serán la nueva población de la siguiente generación. La estrategia antes descrita se repite hasta que se completa el número de iteraciones preestablecido o se resuelve la tarea.

7.1.1 Codificación

La codificación particular hecha para resolver el problema de estabilización simultánea, supone proponer controladores exactamente propios, como individuos de la población. Bredemann [5] demostró que dado un conjunto de m plantas propias, estas son simultáneamente estabilizadas si y solo si las m plantas son simultáneamente estabilizables con un controlador exactamente propio. No existe alguna otra restricción respecto de los individuos de la población, de manera que los polos y ceros del mismo se pueden localizar en cualquier parte del plano complejo, mientras se conserve que los polos o ceros complejos existan en pares conjugados. Es posible desde luego, proponer controladores con grado relativo diferente de cero y hacer variaciones limitando las regiones del plano complejo donde se desea que se localicen los polos y ceros del controlador.

Sea el conjunto de plantas $P = \{P_1, P_2, ..., P_m\}$ con $P \subset \Re(s)$ y la factorización polinomial coprima de cada planta: $P_i = \frac{N_i}{D_i}$ donde $N_i, D_i \in \Re[s]$.

Cada una de las plantas se expresa como:

$$P_i = K_i \frac{\prod_{j=1}^{\deg(N_i)} \left(s - z_{ij}\right)}{\prod_{j=1}^{\deg(D_i)} \left(s - p_{ij}\right)} \text{ donde } K_i \in \Re;\ z_{ij},\ p_{ij} \in \mathbf{C}$$

La población que llamaremos W, se define como el conjunto de individuos (controladores): $W = \{W_1, W_2, ..., W_a\}$, donde a es el número de individuos de la población (parámetro constante del algoritmo), el individuo i es: $W_i = WK_i \frac{WN_i}{WD_i}$ donde: $WK_i \in \Re;\ WN_i,\ WD_i \in \Re[s]$ entonces el conjunto de numeradores WN se establece como $WN = \{WN_i \mid i = 1, 2, ..., a\}$, el conjunto de denominadores $WD = \{WD_i \mid i = 1, 2, ..., a\}$ y el conjunto de ganancias de alta frecuencia $WK = \{WK_i \mid i = 1, 2, ..., a\}$.

Cada individuo se factoriza como:

$$W_i = WK_i \frac{\prod_{j=1}^{b} (s - XN_{ij})}{\prod_{j=1}^{b} (s - XD_{ij})} \text{ donde : } WK_i \in \Re;\ XN_{ij},\ XD_{ij} \in \mathbf{C}$$

b es el tamaño del individuo (parámetro constante del algoritmo)

Se define entonces la población $\overline{W}$ como la terna $\overline{W} = \left(\overline{WN}, \overline{WD}, \overline{WK}\right)$ en el que la matriz de numeradores es: $\overline{WN} = [XN_{ij}]\ \forall i = \{1, 2, ..., a\}, \forall j = \{1, 2, ..., b\}$; la matriz de denominadores es: $\overline{WD} = [XD_{ij}]\ \forall i = \{1, 2, ..., a\}, \forall j = \{1, 2, ..., b\}$; y el vector de ganancias es: $\overline{WK} = [WK_i]\ \forall i = \{1, 2, ..., a\}$.

La terna población original $\overline{W}$ se genera con valores aleatorios con una distribución uniforme en los intervalos definidos por la magnitud máxima de los polos o ceros de las plantas en base a las siguientes restricciones:

$$(-1) \max\left\{\left|z_{ij}\right|\right\} \leq Re\{XN_{ij}\},\ Im\{XN_{ij}\} \leq \max\left\{\left|z_{ij}\right|\right\}$$

$$(-1)\max\left\{\left|\underset{-ij}{p}\right|\right\} \leq Re\{XD_{ij}\},\ Im\{XD_{ij}\} \leq \max\left\{\left|\underset{-ij}{p}\right|\right\}$$

Además de cumplir con que cada elemento del individuo (XN_{ij}, XD_{ij}) tenga su conjugado, esto para que $WN_i,\ WD_i \in \Re[s]$.

La terna de pasos Λ se define como: $\Lambda = (\Lambda N, \Lambda D, \Lambda K)$; teniendo una distribución uniforme tal que:

$$\Lambda N = [\Lambda N_{ij}]\,; \Lambda D = [\Lambda D_{ij}]\,; \Lambda K = [\Lambda K_i]$$

$$Re\{\Lambda N_{ij}\}, Im\{\Lambda N_{ij}\}, Re\{\Lambda D_{ij}\}, Im\{\Lambda D_{ij}\}, \Lambda K_i \in \left[-\frac{1}{2}, \frac{1}{2}\right]$$

$$\forall i = \{1, 2, ..., a\}, \forall j = \{1, 2, ..., b\}$$

La media de los pasos $\mu = (\mu N, \mu D, \mu K)$ se define como:

$$\mu N_j = \sum_{i=1}^{a} \frac{\Lambda N_{ij}}{a};\quad \mu D_j = \sum_{i=1}^{a} \frac{\Lambda D_{ij}}{a};\quad \mu K = \sum_{i=1}^{a} \frac{\Lambda K_i}{a}.$$

La terna $\xi = (\xi N, \xi D, \xi K)$,con: $\xi N_i = (\mu N_i)(\gamma N_i), \xi D_i = (\mu D_i)(\gamma D_i), \xi K = (\mu K)(\gamma K)$, donde la terna aleatoria $\gamma = (\gamma N, \gamma D, \gamma K)$ tiene distribución uniforme cumpliendo que los vectores $\gamma N, \gamma D \in \left\{\frac{2}{3}, \frac{3}{2}\right\}^b$ y el escalar $\gamma K \in \left\{\frac{2}{3}, \frac{3}{2}\right\}$[1].

$\varsigma = (\varsigma N, \gamma D, \varsigma K)$ con: $\varsigma N, \varsigma D \in \mathbb{C}^b$ vectores con distribución normal sobre el plano complejo y $\varsigma K \in \Re$ un escalar con distribución normal sobre los reales.

$R = (RN, RD, RK)$ vector con distribución uniforme; $RN, RD \in \{1, 2, 3, ..., a\}^b$y $RK \in \{1, 2, ..., a\}$, esta terna determina cuales padres, y que elementos de estos, contribuyen a la formación de cada uno de los hijos.

El hijo O queda entonces definido por:

$$O = K_O \frac{N_O}{D_O} = (XK_{RK} + \varsigma K \cdot \xi K) \cdot \frac{\prod_{j=1}^{b}\left[s - \left(XN_{RN_jj} + \varsigma N_j \cdot \xi N_j\right)\right]}{\prod_{j=1}^{b}\left[s - \left(XD_{RD_jj} + \varsigma D_j \cdot \xi D_j\right)\right]} \tag{7.1}$$

En el algoritmo utilizado se han propuesto varias modificaciones de las estrategias evolutivas

[1] Este es un método probado que nos permite cambiar la longitud del paso de manera efectiva [28].

presentadas por Rechenberg [28]:

- Una de ellas consiste en una étapa de mutación generalizada controlada por un parámetro e, que se lleva a efecto si la mejor aptitud de la población se mantiene constante durante e iteraciones. La mutación generalizada consiste en sumar un vector aleatorio tanto a la población como a los pasos, los vectores aleatorios siguen una distribución uniforme acotada por los valores máximos de los polos, ceros y ganancia de los controladores de la población al tiempo en que se decide aplicar la mutación. Esta operación mostró ser útil para la solución de algunos problemas experimentales. Las ternas aleatorias para la mutación generalizada están definidas de la siguiente manera:

  La terna $W' = (WN', WD', WK')$ esta compuesta de vectores aleatorios sobre los reales que se suman a la población cuando el proceso de mutación generalizada se presenta y $\Lambda' = (\Lambda N', \Lambda D', \Lambda K')$ está compuesta por vectores aleatorios sobre los reales que se suman a los pasos, y son tales que:

$WN' = [WN'_{ij}]$: matriz de tamaño a x b con:
$WN'_{ij} \in [-\max\{|XN_{ij}|\}, \max\{|XN_{ij}|\}]\ \forall i, j$
$WD' = [WD'_{ij}]$: matriz de tamaño a x b con:
$WD'_{ij} \in [-\max\{|XD_{ij}|\}, \max\{|XD_{ij}|\}]\ \forall i, j$
$WK' = [WK'_i]$: es un vector de tamaño a con:
$WK'_i \in [-\max\{|WK_i|\}, \max\{|WK_i|\}]\ \forall i$
$\Lambda N' = [\Lambda N'_{ij}]$: matriz de tamaño a x b con:
$\Lambda N'_{ij} \in [-\max\{|\Lambda N_{ij}|\}, \max\{|\Lambda N_{ij}|\}]\ \forall i, j$
$\Lambda D' = [\Lambda D'_{ij}]$: matriz de tamaño a x b con:
$\Lambda D'_{ij} \in [-\max\{|\Lambda D_{ij}|\}, \max\{|\Lambda D_{ij}|\}]\ \forall i, j$
$\Lambda K' = [\Lambda K'_i]$: es un vector de tamaño a con:
$\Lambda K'_i \in [-\max\{|\Lambda K_i|\}, \max\{|\Lambda K_i|\}]\ \forall i$
Todos ellos con distribución uniforme.

- Existe un vector de pasos para cada uno de los individuos de la población, es decir, existe un paso para cada uno de los elementos que conforma el individuo que se trata de un número complejo.

- El mejor individuo de la población junto con su paso, se conserva siempre, aún después de la mutación generalizada.

7.1.2 Función objetivo

Si se tienen m plantas $P_i = \frac{N_i}{D_i}\ \forall i = 1, 2, 3, ...m.$ y un controlador $C = \frac{N_C}{D_C}$ con $N_i, D_i, N_C, D_C \in \Re[s]\ \forall i$, entonces la ES optimiza iterativamente la población en función de las raíces de los polinomios $h_i = N_i N_C + D_i D_C$, buscándose que todos los valores s_0 tales que $h_i(s_0) = 0$ se localicen en el semiplano complejo izquierdo y que el controlador encontrado sea estable (estabilización fuerte simultánea). Si el algoritmo encuentra un controlador C que estabilize simultáneamente las m plantas entonces tendrá una aptitud igual a 1. En este sentido el algoritmo no busca una optimización del controlador en función de sus características de respuesta que tendrá en lazo cerrado con cada una de las plantas. Sin embargo, es posible redefinir la función objetivo presentada para que se alcancen otros parámetros de diseño.

La función objetivo F de la ES se propone como:

$$F(C, P) = \sigma(D_C) \sum_{i=1}^{\sharp(P)} \frac{\Psi(N_i N_C + D_i D_C)}{\sharp(P)}$$

donde $\sharp(P)$ es el número de plantas P a estabilizar.

$$\sigma(D_C) = \begin{cases} 1 & si \quad D_C \text{ es estable} \\ \epsilon & otro\ caso \quad ;\epsilon \approx 0 \end{cases}$$

F asigna un número real entre 0 y 1 para un conjunto de plantas P y un controlador C.

Si se tiene que X es un polinomio real, éste se expresa por la factorización:

$$X = \lambda \cdot \prod_{i=1}^{\deg(X)} (s - x_i)\ \ con\ x_i \in \mathbb{C}\ y\ \lambda \in \Re$$

Haciendo el cambio de variable: $z_i = Re\{x_i\}$, se define Ψ como:

$$\Psi(X) = \sum_{j=1}^{\deg(X)} \left[\frac{[1 - \tanh(z_i - \varepsilon)] \cdot [sgn(z_i) + \delta(x_i)] + sgn(-z_i)}{\deg(X)} \right]$$

donde:

$$sgn(x) = \begin{cases} 1 & si \quad x > 0 \\ 0 & si \quad x \leq 0 \end{cases} ; \delta(x) = \begin{cases} 1 & si \quad x = 0 \\ 0 & si \quad x \neq 0 \end{cases}$$

con ε un numero real casi cero pero positivo.

Nótese que: $\lim_{z_i \to 0^+} (1 - \tanh(z_i)) = 1$.

Dado que z_i son las raíces de los polinomios $h_i = N_i N_C + D_i D_C$, un controlador se considera más apto mientras más se aproximen las raíces con parte real positiva de los polinomios h_i al semiplano complejo izquierdo, el valor de 1 se alcanza cuando todos los h_i son estables y el controlador es estable, esto es, el controlador estabiliza simultáneamente al conjunto de plantas P.

Bibliografía

[1] Abdallah C. T., Dorato P., Bredemann M., New Sufficient Conditions for Strong simultaneous Stabilization, *Automatica*, 33, 1997, pp. 1193-1196

[2] Back, T., *Evolutionary Algorithms in Theory and Practice*, New York Oxford, Oxford University Press, 1996.

[3] Blondel, V., *Simultaneous Stabilization of Linear Systems, Lecture Notes in Control and Information Sciences*, Berlin, Springer-Verlag, 1993.

[4] Blondel, V., et al., Simultaneous stabilization of three or more plants: conditions on the positive real axis do not suffice. *SIAM Journal on Control and Optimization.* 32(2), 1994 pp. 572-590.

[5] Bredemann, M., *Feedback Controler Design for Simultaneous Stabilization*, Ph.D thesis, , Albuquerque New Mexico 87121, University of New Mexico, May 1995.

[6] Chipperfield, A.J., Fonseca, C. M, and Fleming, P.J., Development of genetic optimisation tools for multi-objetive optimisation problems in CACSD. *IEE Colloquium on Genetic Algorithms for Control Systems Engineering Digest*, London WC2R 0BL, Vol. 106, 1992, pp. 3/1-3/6

[7] Chen, H. B., Chow, J. H., Kale, M. A., and Minto, K. D, Simultaneous stabilization using stable system inversion. *Automtica*, 31, 1995, pp 531-542.

[8] Ding, X.., Frank, P.M., Fault detection via factorization approach, *Syst. Cont, Lett.*, 14, 1990, pp 431-436.

[9] Fernández-Anaya, G., Muñoz-Gutiérrez, S., Sánchez-Guzmán, R.A., and Mayol-Cuevas, W.W, Simultaneous stabilization using evolutionary strategies, *International Journal of Control*, 68 (6), 1997, pp. 1417-1435.

[10] Forrest, S., Genetic Algorithms: Principles of Natural Selection Applied to Computation, *Science*, Vol. 261,1993, pp. 872-878.

[11] Goldberg, D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*, (Addison Wesley, Reading, MA), 1989.

[12] Hoffmeister, F., Back, T, *Genetic Algorithms and Evolution Strategies: Similarities and Differences, Parallel Problem Solving from Nature, Proc. 1st Workshop PPSN*, pp. 447-461 Springer, Berlin, Lecture Notes in Computer Science, Vol. 496, , 1991.

[13] Holland, J.P., *Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, Cambridge, Massachusetts, U.S.A., MIT Press, 1992.

[14] Howitt, G. D., and Luus, R., Control of a collection of linear systems by linear state feedback control. *International Journal of Control*, 58, 1993, 79-96.

[15] Kale, M. A., Chow, J. H., and Minto, K. D, A controller parametrization and pole-placement design for simultaneous stabilization. *Proceedings of the 1990 American Control Conference*, 116-121., 1990

[16] Koza J. R., *Genetic Programming*. Cambridge y Londres: The MIT Press. 1992

[17] Koza J. R., *Genetic Programming II. Automatic Discovery of Reusable Programs*. Cambridge y Londres: The MIT Press. 1994.

[18] Li, Y., *Applied Neural Networks IV-Part I: Evolutionary Computing. Lecture Notes (1JYX)*, Departament of Electronics and Electrical Engineering, University of Glasgow, U.K, 1995.

[19] Li,Y., Ng K. C., Murray-Smith D. J., Gray G. J., and Sharman K. C., Genetic algorithm automated approach to the design of sliding mode control systems. *International Journal of Control*, 63 (4), 721-739, 1996

[20] Laboratorio de INvestigación para el Desarrollo Académico, División de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad Nacional Autónoma de México. http://132.248.59.55

[21] Michalewicz, Z., *Genetic Algorithms, Numerical Optimization and Constraints*, Proceedings of the 6th International Conference on Genetic Algorithms, University Pittsburgh, julio 15-19, 1995, Morgan Kaufmann, San Francisco, 1995, pp. 151-158.

[22] Muñoz, S. Fernández, G. Sánchez, R.A. Mayol, W.W. 1997. Evolutionary Algorithms for Plants Simultaneous Stabilization. *IASTED Int. Conf*. Cancún, México. 303-306.

[23] Ng, K. C., and Li, Y., Design of sophisticated fuzzy logic controllers using genetic algorithms. *Proceedings of the 3rd IEEE International Conference on Fuzzy Systems* IEEE World Congress on Computational Inteligence. Orlando, Florida, U.S.A., 1994, pp. 1708-1712.

[24] Ogata K., *Ingeniería de Control Moderna*, Prentice Hall, 1993.

[25] Pascota, M.,et al., Optimal simultaneous stabilization of linear single-input systems via linear state feedback control. *International Journal of Control*, 60(4), 1994, 483-498.

[26] Peña A.,*Desarrollo en Java de una Máquina de Programación Genética*, Tesis de Licenciatura, Facultad de Ingenieria UNAM, México, 1997.

[27] Powell M. J.D., A Survey of Numerical Methods for Unsconstrained Optimization, *on Perspectives on Optimization: A collection of Expository Articles, Addison-Wesley, L.A. California 1972.*

[28] Rechenberg, I., Evolution and Artificial Intelligence, *Machine Learning Principle and Tecnics,*(Forsyth, R.: Chapman and Hall Computing), 1989.

[29] Rechenberg, I., Cybernetic solution path of an experimental problem, *Royal Aircraft Establishment Transl.*, No. 1122, B.F. Toms, Transl. (Ministry of Aviation, Royal Aircraft Establishment, Farnborough, Hants.,United Kingdom, 1965.

[30] Sánchez, R.A., Mayol, W.W., and Figueroa, J.G., Self evolving techniques for representation formation and synthesis in neural networks. *IEEE/AMCA Neural Net-*

works Applied to Control and Image Processing (NNACIP'94). CINVESTAV-IPN, Ciudad de México., 1994, pp 80-85.

[31] Schwefel, and Rudolph, G, *Contemporary evolution strategies. In Advances in Artificial Life*. Berlin Springer, 686-695, 1995

[32] Scilab, free software at INRIA, http://arikara.inria.fr/www-rocq.inria.fr/scilab/

[33] Srinivas, M., and Patnaik, L. M., Genetic algorithms a survey. *IEEE Computer*, 27, 1994, pp. 17-26.

[34] Yao, Y. X., Darouach, M., and Schaefers, J., Simultaneous observation of linear systems. I*EEE transactions on Automatic Control*, 40, 696-699, 1995.

[35] Yun, L., et al., Genetic algorithm automated approach to the design of sliding mode control systems, *International Journal of Control*, 63(4), 1996, 721-739.