UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO

FACULTAD DE INGENIERÍA ELÉCTRICA

ANÁLISIS DE SISTEMAS DINÁMICOS UTILIZANDO HERRAMIENTAS DE INTELIGENCIA ARTIFICIAL

TESIS

Que para obtener el grado de
DOCTORADO EN CIENCIAS DE LA INGENIERÍA ELÉCTRICA

presenta

Julio Agustín Barrera Mendoza

Juan José Flores Romero
Director de Tesis

Julio 2008

A mi a abuela, de quien conservo mucho más que la mitad de mi nombre.

Resumen

El análisis de sistemas dinámicos es de gran importancia en casi todas las áreas de la ciencia. Una forma de análisis de sistemas dinámicos es realizando el trazo del diagrama de bifurcación del sistema dinámico relacionado a un parámetro del mismo. Los métodos para realizar el trazo de diagramas de bifurcación necesitan de valores de inicio y ajuste de parámetros para su correcto funcionamiento, los cuales regularmente no se conocen y requieren de un conocimiento detallado del sistema o de una búsqueda no sistemática de valores para éstos.

Esta tesis muestra el uso de algoritmos de inteligencia artificial como complemento de los métodos para el trazo de diagramas de bifurcación, para el análisis de sistemas dinámicos. Con el fin de facilitar el uso de los métodos para el trazo de diagramas de bifurcación, se han evaluado los algoritmos evolutivos para la búsqueda automatizada de puntos fijos de sistemas dinámicos. También se propone un método para el trazo de diagramas de bifurcación basado en algoritmos evolutivos para entornos dinámicos. Los algoritmos evolutivos no son sensibles a características de las funciones utilizadas para modelar sistemas dinámicos, como la diferenciabilidad, las cuales afectan el desempeño de los métodos numéricos estándar.

Además, se desarrolló un algoritmo para optimización multimodal basado en el algoritmo incremental de envoltura convexa. Éste no necesita de la inicialización de parámetros cuyos valores no son conocidos a priori, como el radio en los algoritmos evolutivos, y que puede afectar su desempeño. Dentro del mismo desarrollo del algoritmo basado en la envoltura convexa se presenta una representación para las facetas y bordes que reduce el uso de recursos de memoria, y permite la implementación de un algoritmo incremental de envoltura convexa que puede utilizarse en cualquier dimensión sin ningún cambio.

Abstract

The analysis of dynamical systems is of great importance in almost all fields of research. One of the analysis tools for dynamical systems is the plot of the bifurcation diagram of the dynamical system related to one parameter of the system. The methods for plotting the bifurcation diagrams require initial values and the setup of parameters that frequently are not known; setting up those parameters require a detailed knowledge of the system being analyzed. Scientist typically use a non-systematic search procedure to determine the values for the parameters.

This thesis shows the use of artificial intelligence algorithms as complements of the methods for plotting bifurcations diagrams, for the analysis of dynamical systems. To facilitate the use of the methods for plotting bifurcation diagrams, evolutionary algorithms are evaluated in the automated search of fixed points of dynamical systems. Also, a method is proposed for plotting bifurcation diagrams based in evolutionary algorithms for dynamical environments. Evolutionary algorithms are not sensitive to characteristics of the functions being analyzed, like differentiability, that affect the performance of standard numerical methods.

Furthermore, an algorithm is presented for multimodal optimization based on the incremental convex hull algorithm. This algorithm does not need the initialization of parameters whose values are not known a priori, like the radius in evolutionary algorithms that can affect its performance. In the same development of the algorithm based on convex hulls a representation is presented for facets and ridges that reduce the use of memory resources, and allows a general implementation of the incremental convex hull algorithm in any dimension.

Contenido

Lista de Figuras

Lista de Tablas

Lista de Algoritmos

Lista de Símbolos

$\mathbb{R}$	Campo de los números reales
F, f	Función de valores reales
x_i	Variable real
$\dot{x}_i$	Primera derivada con respecto del tiempo de la variable x_i
α, β	Parámetros de un sistema dinámico
α_c	valor crítico del parámetro α
$\boldsymbol{\alpha}$	Vector de parámetros de un sistema dinámico
A	Matriz
I	Matriz identidad
$Det(A)$	Determinante de una matriz
γ	Parámetro de amortiguamiento en el algoritmo de Levenberg-Marquardt
λ	Valor propio asociado a una matriz
$Re(\lambda)$	Parte real de un número imaginario
M	Una hipercurva en $\mathbb{R}^n$
y_0, x_0, α_0	Valores iniciales para variables reales
a, b	Individuos de una población
C_j	Valor de aptitud del individuo j-ésimo
P_j	Probabilidad de selección del individuo j-ésimo
M	Número de individuos en una población
N	Número de individuos a seleccionar de una población
p_c	Índice en una secuencia de carácteres
F_c	Parámetro del algoritmo de selección universal estocástica
F_0	Valor inicial para el parámetro F_c
X_s	Conjunto de individuos dominantes
$d(x, y)$	Distancia euclidiana entre los puntos x y y
r	Parámetro del radio
χ	Parámetro de constricción
C_1, C_2	Constantes de apredizaje
R_1, R_2	Valores aleatorios en el rango de $[0, 1]$
ϕ	Suma de las constantes de aprendizaje
v	Velocidad de una partícula
p	Posición de una partícula
p_b	Posición con el mejor valor de aptitud registrado por la partícula

p_g	Posición de la partícula con el valor óptimo global
v_n	Velocidad calculada para una partícula
p_n	Posición calculada para una partícula
CH	Envoltura convexa
$[v_1, v_2, \ldots, v_n]^T$	Un vector en $\mathbb{R}^n$

Lista de Publicaciones

Julio Barrera, Juan J. Flores, and Claudio Fuerte-Esquivel, *"Plotting of Complete Bifurcation Diagrams Using Dynamic Environment Particle Swarm Optimization Algorithm"*, The 2008 International Conference on Artificial Intelligence (ICAI'08: July 14-17, 2008, WorldComp 08)

Julio Barrera, Juan J. Flores, and Claudio Fuerte-Esquivel, *"Generating Complete Bifurcation Diagrams Using a Dynamic Environment Particle Swarm Optimization Algorithm"*, Journal of Artificial Evolution and Applications, vol. 2008, Article ID 745694, 8 pages, 2008. doi:10.1155/2008/745694

Juan J. Flores, Julio Barrera, Félix Calderón, *"Multimodal Optimization by Decomposition of the Search Space in Regions"*, isda, pp. 863-868, Seventh International Conference on Intelligent Systems Design and Applications (ISDA 2007), 2007

Julio Barrera, Juan J. Flores, *"Search of Initial Conditions for Dynamic Systems using Intelligent Optimization Methods"*, cerma, pp. 348-353, Electronics, Robotics and Automotive Mechanics Conference (CERMA 2007), 2007.

Capítulo 1

Introducción

Un sistema dinámico es representado como un conjunto de ecuaciones que especifican cómo las variables cambian a través del tiempo. El conjunto mínimo de variables que determinan de manera única el estado de un sistema son llamadas variables de estado. Las ecuaciones del sistema especifican como calcular nuevos valores de las variables de estado como una función de sus valores actuales y los valores de los parámetros de control. Si permitimos que los parámetros de control cambien, el sistema cambiará con éstos. Si estos parámetros cambian más allá de ciertos valores, el sistema exhibe cambios cualitativos en su comportamiento. Estos cambios cualitativos son llamados bifurcaciones, y los valores de los parámetros donde estos cambios ocurren, son llamados puntos de bifurcación [Kuznetsov98, Strogatz00].

Si permitimos que un parámetro cambie de valor, y graficamos la norma del vector de las variables de estado para las cuales encontramos los puntos fijos del sistema, contra el parámetro que cambia, obtenemos un diagrama al que llamamos diagrama de bifurcación. En un diagrama de bifurcación podemos ver que los puntos fijos pueden desaparecer, aparecer o cambiar su estabilidad, conforme el parámetro cambia de valor. Estos cambios pueden ocurrir incluso al variar el valor del parámetro infinitesimalmente. Los diagramas de bifurcación son herramientas utilizadas en el análisis de estabilidad de sistemas dinámicos.

La construcción tradicional de un diagrama de bifurcación se lleva a cabo con los llamados métodos de continuación; estos métodos necesitan como punto inicial, un punto fijo estable del sistema que se analiza. Si no se tiene información del sistema que se analiza, encontrar un punto fijo estable de un sistema no es fácil; éstos pueden ser encontrados mediante simulación, iniciar con un punto cercano a un punto estable, y mediante las ecuaciones que

modelan el sistema, seguir la posición del punto con respecto al cambio del tiempo, cuando la posición del punto no presente cambios en su posición con respecto al tiempo, el punto en esa posición fija se considera un punto fijo estable del sistema. Otro método para encontrar un punto fijo estable es igualar las ecuaciones que modelan el sistema dinámico a cero y encontrar una solución al sistema de ecuaciones [Ermentrout06, Strogatz00].

Ambos métodos tienen desventajas al obtener puntos fijos tienen desventajas; si nuestro punto inicial en la simulación no es lo suficientemente cercano a un punto estable, este puede seguir cambiando su posición con respecto al tiempo de forma indefinida [Ermentrout06]. Aunque existe una gran variedad de métodos para encontrar soluciones de sistemas de ecuaciones no lineales [Kincaid91], éstos dependen de la selección de un punto inicial adecuado. Además es necesario que el sistema de ecuaciones presente características como continuidad y diferenciabilidad de las ecuaciones.

Una vez obtenido un punto fijo del sistema, el método de continuación incrementa uno de los parámetros de control del sistema y calcula el cambio de posición del punto fijo con respecto al cambio del parámetro [Kuznetsov98]. Los métodos para calcular este cambio en la posición están basados en los métodos para encontrar soluciones de sistemas de ecuaciones no lineales, los llamados métodos de gradiente [Govaerts00]. El punto fijo estable del sistema es utilizado como punto inicial para estos métodos; es indispensable que el punto inicial sea estable de otra forma el método no funcionará [Ermentrout06].

Al ser semejantes a los métodos para el cálculo de soluciones de ecuaciones no lineales, los métodos de continuación presentan las mismas desventajas [Kuznetsov98]. Es necesario que las ecuaciones que modelan al sistema sean continuas y diferenciables; además presentan un conjunto de parámetros, (relacionados con el método en si, no con los parámetros del sistema) que deben ser inicializados adecuadamente o es posible que no se obtenga el diagrama de bifurcación [Ermentrout06].

Por otra parte no es posible hacer el cálculo del cambio de la posición del punto inicial variando más de un parámetro al mismo tiempo; un análisis multi-paramétrico debe ser realizado variando un sólo parámetro a la vez y fijando los valores de los demás parámetros.

1.1. Objetivos

El objetivo de la tesis es efectuar un análisis de los diferentes algoritmos evolutivos y de su aplicabilidad para encontrar soluciones a sistemas de ecuaciones no lineales como aplicación al análisis de sistemas dinámicos. Así mismo proporcionar una herramienta alternativa a los programas estándar xppaut [Ermentrout06] y AUTO [Doedel97] para el trazo de diagramas de bifurcación utilizando técnicas de optimización inteligente. Además diseñar un método de optimización inteligente que tenga características que no se encuentren en los algoritmos de optimización actuales, como es el no necesitar de un parámetro de radio en algoritmos genéticos [Li J.-P.02] y optimización de enjambre de partículas [Parrott06].

1.2. Tesis

Es posible utilizar algoritmos de inteligencia artificial para la búsqueda de puntos fijos de sistemas sistemas dinámicos. Además se pueden generar diagramas de bifurcación completos, variando más de un parámetro, utilizando algoritmos de inteligencia artificial.

1.3. Justificación

Al iniciar el estudio de un sistema dinámico si no se tiene información previa de éste, incluso el encontrar un punto fijo estable del mismo es una tarea difícil. Hacer una simulación o utilizar métodos para búsqueda de soluciones de sistemas de ecuaciones no lineales, como el método de Newton o el método de la secante [Kincaid91], no garantiza encontrar puntos fijos en un tiempo corto. Además los métodos de Newton y de la secante requieren de inicialización. Sin información, la búsqueda puede convertirse en una búsqueda aleatoria y consumir mucho tiempo; además es posible que el punto que se encuentre no sea el adecuado de acuerdo a las condiciones del sistema dinámico que se analiza, siendo necesario realizar más de una búsqueda. No existen métodos basados en gradiente o simulación que permitan de manera general encontrar soluciones múltiples de un sistema de ecuaciones o más de un punto estable en el caso de simulación.

En el proceso de análisis de sistemas dinámicos es necesario contar con una herramienta de búsqueda automatizada que permita obtener más de un punto estable, o que tengan propiedades que puedan mejorar el desempeño de métodos basados en gradiente, sin

necesitar la supervisión del investigador. Una herramienta con estas características permitiría ahorrar tiempo y trabajo al investigador en la inicialización de métodos de gradiente, tiempo que podría ser utilizado en el análisis del sistema. Además el tener más de un punto fijo permite tener más de un punto inicial para los métodos de continuación y posiblemente permitir un mejor análisis de un sistema dinámico.

Las implementaciones del método de continuación no son simples de utilizar. Para un investigador principiante es necesario reservar tiempo para el aprendizaje de la interfaz, y aun para un investigador familiarizado con la implementación, para cada nuevo sistema dinámico a analizar es necesario adecuar los parámetros de la implementación para que ésta funcione correctamente.

Otro punto a considerar es el análisis multi-paramétrico, con las implementaciones que existen del método de continuación, como es xppaut y AUTO, no es posible hacerlo de forma directa. Para el caso de dos parámetros, por ejemplo, es necesario fijar un parámetro y generar un diagrama de bifurcación para el parámetro restante, después variar el parámetro que fue fijado inicialmente y generar otro diagrama de bifurcación completo. Este procedimiento se repite para generar una superficie y debe ser hecho completamente por el investigador ya que la implementación no cuenta con automatización en este nivel.

También es deseable simplificar el proceso de generación de diagramas de bifurcación. Aunque no es posible eliminar del todo los parámetros en la implementación de un método para generación de diagramas de bifurcación, al reducir el número de éstos se simplifica el proceso y ahorra tiempo.

En la generación de diagramas de bifurcación multi-paramétricos además de la automatización sería conveniente poder cambiar más de un parámetro a la vez. Como se ha dicho antes con las implementaciones actuales sólo se puede variar un parámetro a la vez y los restantes deben permanecer fijos.

1.4. Estado del arte

El estudio de los sistemas dinámicos es extenso tanto en la teoría [Kuznetsov98, Seydel99, Strogatz00] como en la implementación de métodos numéricos [Eusebius97, Doedel97, Ermentrout06, Govaerts00, Richter83, Seydel94]. De manera general los métodos de continuación aun permiten sólo la variación de un parámetro a la vez. Las implementaciones más utilizadas [Doedel97, Ermentrout06] no ha presentado cambios en los últimos años.

El uso de herramientas de inteligencia artificial para el análisis de sistemas dinámicos ha sido tratado anteriormente, v.g. el trabajo de Elisha Sacks, en el cual se logra determinar el diagrama de fase de un sistema de dos ecuaciones lineales con un parámetro [Sacks91].

Hasta donde el autor conoce no existen a la fecha referencias de trabajos previos en el análisis de sistemas dinámicos mediante el trazo de diagramas de bifurcación utilizando algoritmos de optimización inteligente; no obstante existen trabajos previos en la búsqueda de soluciones de sistemas de ecuaciones no lineales. Dichos trabajos son los realizados por Varum Aggarwal [Aggarwal00] y Crina Grosan [Crina Grosan06].

Los algoritmos genéticos han sido usados en una gran variedad de problemas de optimización [Randy L. Haupt04]; la característica principal de los algoritmos genéticos es su convergencia a un óptimo global, incluso en problemas donde los algoritmos tradicionales no convergen. Esta característica no es deseable cuando necesitamos encontrar más de un óptimo; éste es el caso de los problemas multimodales.

En problemas multimodales la ventaja de los algoritmos genéticos, (su convergencia a un solo óptimo global) se convierte en una desventaja; los individuos de una población tienden a ser muy semejantes a través de las generaciones, a esto se le conoce como perdida de diversidad. La perdida de diversidad no es deseable cuando se resuelve un problema multimodal; es necesario llevar un seguimiento de los individuos que pueden conducir a un óptimo. Así el método principal para resolver problemas multimodales con algoritmos genéticos es evitar la perdida de diversidad [Coello99].

Para preservar la diversidad, un tipo de algoritmos genéticos está basado en la observación de que en la naturaleza muchas especies coexisten en una región geográfica cerrada, compartiendo recursos y creando nichos. Esos algoritmos genéticos son llamados algoritmos de "niching" [Mahfoud95].

Existen dos tipos principales de algoritmos de "niching": "fitness sharing" y "crowding". En los algoritmos de "fitness sharing" se selecciona un individuo con un valor de aptitud alto y el valor de aptitud de individuos semejantes a él son penalizados. La penalización se realiza a través de una función de atenuación, la cual determina si un individuo es suficientemente semejante al individuo con valor de aptitud alto, para ser penalizado. Regularmente la función de atenuación depende de un parámetro que necesita ser inicializado de manera adecuada para que el algoritmo funcione correctamente. El grado de semejanza es evaluado como la distancia entre individuos, y el parámetro para determinar cuales indi-

viduos son afectados por la función de atenuación es llamado radio. El valor del parámetro radio no es conocido a priori; en la mayoría de los casos es necesario determinarlo por medio de un análisis independiente [Li J.-P.02] u otras técnicas [Miller96].

Los algoritmos de "crowding" tratan de preservar la diversidad sustituyendo individuos de una población por sus descendientes, lo cual se realiza mediante la comparación de cada descendiente con un subconjunto de la población seleccionado aleatoriamente, el individuo del subconjunto que tenga mayor semejanza con el descendiente que se esta comparando, será reemplazado por el descendiente. El tamaño del subconjunto aleatorio es determinado por el llamado factor de "crowding". No hay un algoritmo para determinar el factor de "crowding". Una revisión más detallada de los algoritmos de "crowding" puede ser encontrada en el trabajo de Mahfoud [Mahfoud95].

En el ámbito de la optimización multimodal mediante el uso de algoritmos genéticos, en específico en los algoritmos de tipo "niching", los de reciente interés son los algoritmos basados en la separación de la población en especies. El trabajo más representativo es el realizado por J.P. Li [Li J.-P.02]. Algunos trabajos recientes incluyen modificaciones a algoritmos de "fitness sharing"; una referencia completa de los algoritmos actuales puede encontrarse en [Singh06].

La optimización multimodal no se limita a los algoritmos genéticos; existen algoritmos basados en otros modelos naturales como son los algoritmos de enjambre de partículas. Los algoritmos de enjambres de partículas están inspirados en el comportamiento social de las parvadas de aves y los cardúmenes de peces. Estos algoritmos fueron presentados por primera vez por Eberhart y Kennedy [Kennedy95]. Los métodos de optimización de enjambre de partículas comparten algunas características con los algoritmos genéticos; se inicializa una población de puntos al azar y se realiza la búsqueda de un óptimo actualizando la población en cada generación. No obstante, en este algoritmo no existen operadores de cruza ni mutación; cada partícula mantiene la información del óptimo global de la generación, además de la posición en la cual la partícula ha obtenido el mejor valor de aptitud. El concepto principal de la optimización de enjambre de partículas consiste en la actualización de la velocidad de la partícula en la dirección del óptimo global y del mejor valor de aptitud registrado.

Los trabajos de investigación recientes incluyen modificaciones al algoritmo de base de enjambres de partículas. En particular, el uso del método de conservación de especies en el ámbito de enjambres de partículas. V.g. el trabajo de Parrot [Parrott06].

Los algoritmos para optimización multimodal citados requieren de parámetros que deben ser inicializados y cuyos valores no son conocidos a priori. Como ejemplo está el parámetro de radio; la selección de un valor adecuado afecta el desempeño del algoritmo [Li J.-P.02]. En esta investigación doctoral, se propone un algoritmo basado en el algoritmo incremental de envoltura convexa [Franco P. Preparata85], el cual elimina el uso de parámetros como el radio y logra determinar todos los óptimos de una función determinada.

La descripción de los métodos numéricos para el trazo de diagramas de bifurcación puede ser encontrada en los textos de Willy Govaerts [Govaerts00] y Yuri Kuznetsov [Kuznetsov98].

1.5. Metodología

Para realizar el trazo de un diagrama de bifurcación mediante los métodos tradicionales como es el método de continuación y sus variantes [Ermentrout06, Eusebius97, Doedel97, Govaerts00] es necesario contar con un punto fijo estable del sistema dinámico que se analiza. Los puntos fijos de un sistema dinámico pueden obtenerse al igualar todas las derivadas de las ecuaciones que modelan el sistema dinámico a cero, dando como resultado un sistema de ecuaciones que generalmente es no lineal.

Este problema, encontrar las soluciones de un sistema de ecuaciones no lineales, es abordado como un problema de optimización: se busca minimizar la norma del resultado de evaluar un punto del espacio de búsqueda (en este caso, el espacio fase del sistema dinámico) en el sistema de ecuaciones no lineales. Este enfoque no es único, mediante una transformación que preserva los óptimos de una función es posible cambiar el problema de una búsqueda de ceros a una búsqueda de máximos.

Una vez que el problema puede ser expresado como un problema de optimización, se evalúa el desempeño de varios algoritmos de inteligencia artificial utilizados en optimización; los llamados algoritmos evolutivos y el algoritmo del explorador el cuál fue desarrollado en la presente tesis.

El trazo de un diagrama de bifurcación tiene como base el calcular los cambios que presenta un punto fijo de un sistema dinámico con respecto a uno o varios parámetros del sistema dinámico. Los cambios que presenta un punto fijo pueden ser tanto en su posición en el espacio fase, como en la estabilidad del mismo. En este caso solamente se trata el problema de seguir el cambio de posición de un punto fijo, y no así sus posibles cambios en

la estabilidad.

Para resolver el problema del trazo de diagramas de bifurcación se toma el mismo enfoque del problema de encontrar las soluciones de un sistema de ecuaciones no lineales; el problema es visto como un problema de encontrar óptimos, que además al agregar el cambio de la posición que pueden sufrir los óptimos es visto como un problema que dentro del ámbito de optimización es llamado optimización en entornos dinámicos.

Una vez que el trazo de diagramas de bifurcación ha sido planteado como un problema en un entorno dinámico, se utiliza un método de inteligencia artificial, específicamnte la optimización de enjambre de partículas, para buscar las soluciones de un sistema de ecuaciones no lineales relacionado a un sistema dinámico, y seguir los cambios de posición que se presentan al variar uno o más parámetros del sistema dinámico.

1.6. Contenido de la tesis

En este capítulo se ha dado un breve introducción a la presente tesis, así como los objetivos, justificación y el trabajo actual de investigación relacionado a ésta.

En el Capítulo 2 se enuncia el problema del análisis de sistemas dinámicos mediante la generación de diagramas de bifurcación.

En el Capítulo 3 se describen los algoritmos de inteligencia artificial utilizados en la presente tesis.

El Capítulo 4 presenta el "algoritmo del explorador" desarrollado para la búsqueda de óptimos de una función dada basados en el algoritmo de envoltura convexa.

En el Capítulo 5 se describe la metodología que siguió en la realización de la presente tesis, así como los algoritmos utilizados en los casos de estudio.

En el Capítulo 6 se muestran los resultados obtenidos al aplicar los algoritmos de inteligencia artificial y el algoritmo desarrollado en los casos de estudio. Las conclusiones de los resultados obtenidos de la aplicación de los algoritmos de inteligencia artificial son mostradas en el Capítulo 6, así como el planteamiento del desarrollo futuro a partir del presente trabajo.

Capítulo 2

Análisis de Sistemas Dinámicos

En este capítulo se da una breve introducción a sistemas dinámicos, se describe el método estándar para el análisis de sistemas dinámicos mediante el trazo de diagramas de bifurcación, y se enuncian de manera formal los problemas que presenta el trazo de diagramas de bifurcación.

2.1. Definición de sistema dinámico

La noción de un sistema dinámico es la formalización matemática del concepto científico general de un proceso determinista. Los estados pasados y futuros de muchos sistemas físicos, biológicos, químicos, económicos, ecológicos e incluso sociales pueden ser predichos hasta cierto punto si conocemos el estado actual y las leyes que gobiernan su evolución. Si estas leyes no cambian en el tiempo, podemos considerar el comportamiento de tal sistema como completamente definido por su estado inicial. Así, la noción de un sistema dinámico incluye un conjunto de posibles estados y una ley de evolución de un estado en el tiempo.

La palabra bifurcación, que significa dividirse en dos ramas o brazos, es utilizada para describir cualquier situación en la cual la fotografía cualitativa, topológica del objeto que estamos estudiando se altera con el cambio de alguno de los parámetros de los que depende el objeto bajo análisis. Estos objetos puede ser de una gran variedad.

Uno de los métodos de análisis de sistemas dinámicos consiste en determinar los puntos de bifurcación. Éste es realizado regularmente en dos partes: la primera consiste en determinar un punto fijo estable del sistema. Los puntos fijos estables se encuentran

regularmente primero determinando los puntos fijos de un sistema y después determinando la estabilidad del punto fijo encontrado. Este punto estable es después utilizado como punto inicial para el método llamado "de continuación", el cual, como su nombre lo indica, consiste en construir (a partir del punto inicial) un diagrama donde se sigue la evolución del punto con respecto al cambio de un parámetro del sistema.

El presente trabajo se desarrolló con el objetivo de utilizar algoritmos de inteligencia artificial como herramientas auxiliares a los algoritmos numéricos ya existentes para el análisis de sistemas dinámicos; en particular en el trazo de diagramas de bifurcación. En el trazo de diagramas de bifurcación hay dos etapas que son de especial interés: la primera consiste en buscar un punto estable del sistema, que como se ha dicho se realiza con una búsqueda no sistemática por parte del investigador de manera no automatizada y requiere de conocimiento previo del sistema que se está analizando. La segunda es en sí el trazo de diagramas de bifurcación; éste presupone que las funciones que se están analizando son continuas y que no presentan ninguna de las características que pudieran en algún momento influir en el desempeño y comportamiento de los algoritmos utilizados, como pueden ser puntos donde la función no sea diferenciable o ésta no este definida.

En las siguientes secciones se presenta una descripción de los problemas en concreto que se deben de resolver para cada una de las etapas del análisis de sistemas dinámicos, en particular en el trazado del diagrama de bifurcación relacionado con un parámetro del sistema.

2.2. Puntos fijos y estabilidad

Un sistema dinámico puede ser descrito por un sistema de ecuaciones diferenciales de primer grado. Para realizar el análisis de un sistema dinámico se comienza con la búsqueda de un punto estable del sistema. Un punto estable debe cumplir con la condición de que los valores para todas las derivadas deben ser iguales a cero. Ésto es, si nuestro sistema dinámico es descrito por el sistema de ecuaciones mostrado en la Ecuación (2.1)

$$\begin{aligned} \dot{x}_1 &= f_1([x_1, x_2, \ldots, x_n]^T) \\ \dot{x}_2 &= f_2([x_1, x_2, \ldots, x_n]^T) \\ &\vdots \\ \dot{x}_n &= f_n([x_1, x_2, \ldots, x_n]^T) \end{aligned} \tag{2.1}$$

donde $\dot{x}_i$ es la primera derivada con respecto del tiempo de la variable x_i, necesitamos encontrar un vector $[x_1^*, x_2^*, \ldots, x_n^*]^T$ tal que

$$\begin{aligned} 0 &= f_1([x_1^*, x_2^*, \ldots, x_n^*]^T) \\ 0 &= f_2([x_1^*, x_2^*, \ldots, x_n^*]^T) \\ &\vdots \\ 0 &= f_n([x_1^*, x_2^*, \ldots, x_n^*]^T) \end{aligned} \tag{2.2}$$

Este vector no es necesariamente único en la región de operación del sistema dinámico. El sistema de ecuaciones 2.2 es llamado también sistema de ecuaciones algebraicas del sistema dinámico. Debe notarse que no expresamos el tiempo como una de las variables en los sistemas de Ecuaciones (2.1), (2.2). Ésto es, consideramos el tiempo de manera implícita. A los sistemas expresados con tiempo explícito se les llama sistemas autónomos.

El sistema dinámico representado por la Ecuación (2.1) no contiene parámetros. De manera general, los sistemas dinámicos también dependen de parámetros, los cuales son expresados en el sistema de ecuaciones del sistema dinámico, como se muestra en la Ecuación (2.3).

$$\begin{aligned} \dot{x}_1 &= f_1([x_1, x_2, \ldots, x_n]^T, [\alpha, \beta, \ldots]^T) \\ \dot{x}_2 &= f_2([x_1, x_2, \ldots, x_n]^T, [\alpha, \beta, \ldots]^T) \\ &\vdots \\ \dot{x}_n &= f_n([x_1, x_2, \ldots, x_n]^T, [\alpha, \beta, \ldots]^T) \end{aligned} \tag{2.3}$$

De manera general se buscan puntos fijos del sistema de Ecuaciones (2.3) donde $x_1, x_2, \ldots, x_n$ son las variables de estado del sistema dinámico y $\alpha, \beta, \ldots$ son parámetros relacionados al sistema dinámico, los parámetros pueden variar en rangos determinados. En términos de la Ecuación (2.1), los puntos fijos representan puntos de equilibrio. Un punto de equilibrio se considera estable si todas las perturbaciones suficientemente pequeñas que se realicen al sistema terminarán por amortiguarse con el tiempo. Es decir, si tomamos un punto cercano y seguimos su trayectoria con el tiempo, éste eventualmente convergerá al punto fijo. De manera inversa, un punto fijo se considera inestable si una perturbación que se realiza en el punto tiende a aumentar con el tiempo. Al seguir la trayectoria con el tiempo de un punto cercano este se alejara del punto fijo.

La estabilidad de un punto también puede ser determinada sin necesidad de observar el comportamiento de una pequeña perturbación de éste. Por ejemplo las condiciones suficientes para que un punto fijo x_0 sea estable son provistas por el Teorema 1 [Kuznetsov98]

Teorema 1 *(Lyapunov) considere un sistema dinámico definido por*

$$\dot{x} = f(x), \qquad x \in \mathbb{R}^n, \tag{2.4}$$

donde f es una función continua y existe su derivada. Suponga que f tiene un punto fijo x_0 (i.e. $f(x_0) = 0$), y sea A la matriz del Jacobiano de $f(x)$ evaluada en el punto fijo, $A = f_x(x_0)$. Entonces x_0 es estable si todos los valores propios $\lambda_1, \lambda_2, \ldots, \lambda_n$ de A satisfacen $Re(\lambda) < 0$.

Recordando que los valores propios son raíces de la ecuación característica obtenida al calcular el determinante

$$\det(A - \lambda I) = 0, \tag{2.5}$$

donde I es la matriz identidad de $n \times n$.

El teorema puede ser probado de forma simple para un sistema lineal [Kuznetsov98]

$$\dot{x} = Ax, \qquad x \in \mathbb{R}^n \tag{2.6}$$

mediante su solución explícita en una base donde A tiene una forma normal de Jordan, al igual que para un sistema no lineal, linealizando el problema cerca del punto fijo x.

2.3. Bifurcaciones

En la vida diaria observamos cambios que pueden ocurrir gradualmente o de forma brusca. Regularmente clasificamos los cambios como cualitativos o cuantitativos. Por ejemplo, considere una banca con soportes en los extremos y una carga sobre ésta (ver Figura 2.1(a)); si la carga α es pequeña, la banca tomará una forma curva con una deformación que depende de magnitud de la carga α y de las propiedades de los materiales de la tabla (ver Figura 2.1(b)).

Este estado de la tabla permanecerá estable en el sentido de que una pequeña variación en la carga α (o en las propiedades del material) conducirá a un estado ligeramente perturbado. Tal variación es referida como un cambio cuantitativo. La banca es deformada dentro de su régimen de elasticidad y regresará a su forma original cuando la perturbación en α es removida (ver Figura 2.1(c)).

La situación cambia abruptamente cuando la carga α es aumentada. Más allá de un nivel critico α_c, la banca se rompe. Esta acción repentina es un ejemplo de un cambio cualitativo (ver Figura 2.1(d)); además ésto sucede cuando las propiedades materiales son cambiadas mas allá de un límite. Supongamos que la forma de la banca es modelada por alguna función f. Podemos decir, de manera poco formal, que existe una solución de la función f para valores de carga $\alpha < \alpha_c$ y que esta solución deja de existir para $\alpha > \alpha_c$. La carga α es un ejemplo de un parámetro. En un problema práctico es necesario encontrar los valores de los parámetros de un sistema para los cuales ocurre una transición entre un

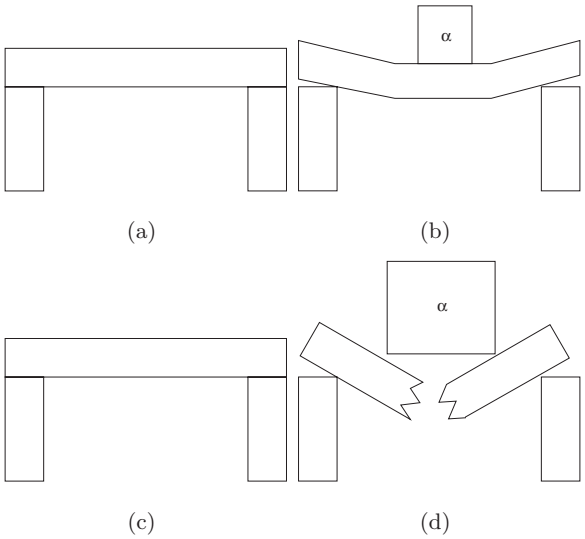


Figura 2.1: Varios estados de un tabla en diferentes condiciones de carga: a) estado estable b) estado con una pequeña perturbación α c) recuperación del estado estable después de remover la perturbación α d) cambio cualitativo al agregar una perturbación α mayor que el límite α_c.

cambio cuantitativo y uno cualitativo; conocer el valor de la carga $\alpha = \alpha_c$ para el cúal la tabla dejará de doblarse para romperse.

De manera formal, considere un sistema dinámico que depende de un parámetro. En el caso de un sistema continuo, con el tiempo lo escribiremos así

$$\dot{x} = f(x, \alpha), \tag{2.7}$$

donde $x \in \mathbb{R}^n$ y $\alpha \in \mathbb{R}^m$ representan las variables de estado y parámetros respectivamente. Considere del diagrama de fase del sistema que representa la Ecuación (2.7); al variar el parámetro el diagrama fase también cambia. Hay dos posibilidades: el sistema puede permanecer topológicamente equivalente al sistema original: ocurre un cambio cuantitativo; o su topología cambia: ocurre un cambio cualitativo.

La aparición de un diagrama de fase no equivalente topológicamente bajo la variación de un parámetro es llamada bifurcación. Así, una bifurcación es un cambio en el tipo de topología del sistema, cuando éste pasa a través de un valor crítico (valor de bifurcación).

2.4. Trazo de diagramas de bifurcación

Consideremos un sistema continuo en el tiempo que depende de un parámetro

$$\dot{x} = f(x, \alpha), \qquad x \in \mathbb{R}^n, \qquad \alpha \in \mathbb{R}^1 \tag{2.8}$$

donde f s una función continua y existe su derivada de (x, α). El trazo de diagramas de bifurcación de un sistema significa la construcción de la gráfica de la Ecuación (2.8). En particular, determinando la dependencia de los puntos estables con respecto de los parámetros, así como la localización y análisis de los puntos de bifurcación.

Los puntos de equilibrio de 2.8 satisfacen

$$f(x, \alpha) = 0 \tag{2.9}$$

La Ecuación (2.9) representa un sistema de n ecuaciones escalares en $\mathbb{R}^{n+1}$ dotado con las coordenadas (x, α). Como se ha mencionado, de manera general, la Ecuación (2.9) define una curva M de dimensión 1 en $\mathbb{R}^{n+1}$. El cálculo de esta curva (llamada de equilibrio) nos da la dependencia de un punto fijo de la Ecuación (2.8) en el parámetro α.

El problema de calcular la curva 1-dimensional M es un caso específico que se resuelve con el método de continuación, y significa encontrar una curva en $\mathbb{R}^{n+1}$ definida por n ecuaciones

$$F(y) = 0, \qquad F : \mathbb{R}^{n+1} \to \mathbb{R}^n \tag{2.10}$$

Utilizando el teorema de la función implícita [Kuznetsov98], la Ecuación (2.10) define una curva suave M que pasa por el punto y_0 y que satisface 2.10, dado que el rango de $J = n$, donde $J = F_y(y_0)$ es la matriz del Jacobiano de la Ecuación (2.10), evaluada en y_0 (condición de regularidad). En la Ecuación (2.9) tenemos $y = (x, \alpha)$ y la condición de regularidad es satisfecha en un punto fijo estable.

La solución numérica del problema de continuación en la Ecuación (2.10) significa calcular una secuencia de puntos

$$y_1,\ y_2,\ y_3, \ldots \tag{2.11}$$

que aproximan la curva M con la precisión deseada. Se considera conocido un punto adicional y_0, que sea suficientemente cercano a M (o que pertenezca a M), desde donde la secuencia pueda ser generada en una de las dos posibles direcciones. En el caso de un punto fijo, el punto $y_0 = (x_0, \alpha_0)$ corresponde usualmente a un punto fijo x_0 de la Ecuación (2.8), encontrado con un valor fijo α_0 del parámetro, por medio de algún método numérico o por búsqueda no sistematizada.

La mayoría de los métodos de continuación usados en el análisis de bifurcación implementan métodos de predicción-corrección, los que incluyen tres pasos básicos que se realizan de forma repetida [Kuznetsov98]

i) Predicción del siguiente punto.

ii) Corrección.

iii) Control del tamaño del paso.

No existen al momento programas de análisis de sistemas dinámicos que permitan de manera general el trazo de un diagrama de bifurcación variando dos parámetros al mismo tiempo. Además de que es necesario que el punto inicial para el trazo del diagrama sea un punto estable del sistema. Esta tesis propone un algoritmo alternativo para el trazo de diagramas de bifurcación sin necesidad de iniciar el trazo desde un punto estable. Además permite la variación de más un de parámetro al mismo tiempo, y producir diagramas de bifurcación más completos, utilizando técnicas de optimización dinámica.

2.5. Conclusiones del capítulo

En este capítulo se ha dada una breve introducción a los sistemas dinámicos y los conceptos básicos relacionados, como es el concepto de estabilidad en puntos fijos. También se define el concepto de bifurcación y se muestran los pasos involucrados en el proceso del trazo de un diagrama de bifurcación. Se muestra que la condición de iniciar el trazo de un diagrama de bifurcación en un punto estable tiene su origen en el marco teórico del estudio de las bifurcaciones.

Capítulo 3

Algoritmos de Inteligencia Artificial

En el presente capítulo se describen los algoritmos de inteligencia artificial utilizados en el desarrollo de esta tesis; en particular los llamados algoritmos evolutivos. Estos algoritmos han sido usados en optimización teniendo éxito en problemas donde los algoritmos estándar se desempeñan de manera pobre o no pueden ser utilizados. Los algoritmos evolutivos no dependen de propiedades de las funciones a optimizar, como son la continuidad o diferenciabilidad.

3.1. Algoritmos genéticos

Los algoritmos genéticos son algoritmos de optimización inspirados en la teoría de evolución de Darwin; los individuos con mejores características para su entorno tienen mejores posibilidades de supervivencia. Estos algoritmos han sido utilizados con éxito en problemas de optimización, incluyendo problemas donde los algoritmos numéricos fallan o no son aplicables. Estos algoritmos proveen una herramienta para la búsqueda automatizada de un óptimo para una función dada. Los algoritmos genéticos son no-determinísticos, *i.e.*, es posible encontrar un óptimo diferente para un mismo problema en cada ejecución.

El algoritmo comienza con una población (un conjunto de individuos) donde cada individuo representa un punto en el espacio de búsqueda. Cada individuo posee un valor de aptitud que sirve para determinar si un individuo es mejor que otro. Este valor de aptitud está dado por la función que se quiere optimizar, llamada función objetivo, y así la población

está dentro del dominio de la función a optimizar. En la representación de los individuos también se toman conceptos de la genética; los individuos (puntos del espacio de búsqueda) son representados por secuencias de caracteres (generalmente se usa la representación binaria). A esta secuencia de caracteres, que representa a un individuo, se le llama cromosoma. El cromosoma de un individuo es decodificado en una representación adecuada, un número de punto flotante, un carácter o cualquier otro tipo de valor del espacio de búsqueda para evaluar el individuo utilizando la función que se quiere optimizar y así obtener su valor de aptitud.

Para encontrar un valor óptimo de la función, dada una población inicial, ésta es modificada en iteraciones. A cada iteración se le llama generación. Para pasar de una generación a la siguiente se aplican operadores a la población; estos operadores son selección, cruza y mutación (en analogía al comportamiento de los seres vivos). El operador de selección selecciona a algunos de los mejores individuos para aparearse; el operador de cruza genera nuevos individuos a partir de los individuos previamente seleccionados; dos nuevos individuos, los descendientes, son creados a partir de dos individuos seleccionados llamados padres. El tamaño de la población es mantenida constante eliminando los peores individuos en la población. El operador de mutación realiza un cambio aleatorio en un individuo; si el individuo representa un punto en un espacio de búsqueda n-dimensional, un cambio aleatorio es introducido en una de sus coordenadas, seleccionada también aleatoriamente. La Figura 3.1 muestra el diagrama de flujo para los algoritmos genéticos.

3.1.1. Cruza

Para ilustra el operador de cruza en algoritmos genéticos, supongamos que dos individuos a y b de una población están representados por las secuencias de 20 caracteres de la Figura 3.2 numerados del 0 al 19

Para realizar la cruza de los individuos a y b tomaremos una parte del cromosoma de a y otra del cromosoma de b para formar la representación de un nuevo individuo c, como todos los individuos deben tener el mismo número de caracteres, para realizar el procedimiento se selecciona primero un punto de cruza p_c al azar, después se toma la subsecuencia a partir del índice 0 hasta el índice p_c del individuo a y la subsecuencia a partir del índice $(p_c + 1)$ hasta 19 de individuo b, al unir estas dos subsecuencias se genera un cromosoma completo de 20 caracteres, así obtenemos la representación de un nuevo

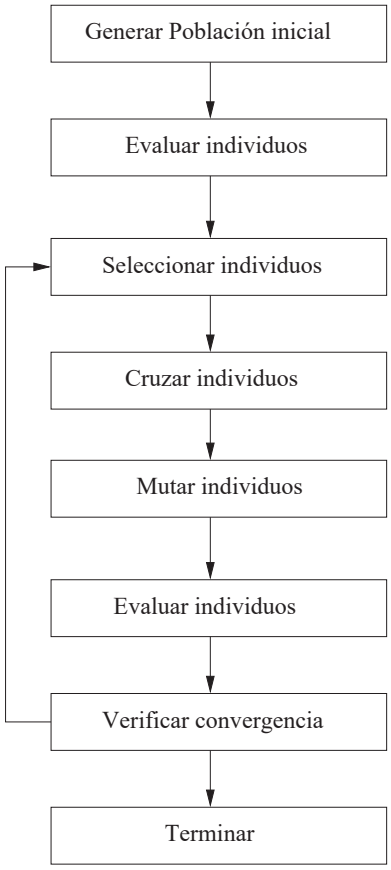


Figura 3.1: Diagrama de flujo para algoritmos genéticos.

$$\begin{aligned} cromosoma(a) &= 01010111000011100000 \\ cromosoma(b) &= 00010101001010010101 \end{aligned}$$

Figura 3.2: Representación de dos individuos mediante una secuencia de caracteres.

individuo, como se ilustra en la figura 3.3

$$
\begin{aligned}
cromosoma(a) &= \overbrace{\mathbf{0101011}}^{0,\cdots,p_c} 1000011100000 \\
cromosoma(b) &= 0001010 \underbrace{\mathbf{1001010010101}}_{(p_c+1),\cdots,19} \\
\\
cromosoma(c) &= \overbrace{0101011}^{cromosoma(a)} \underbrace{1001010010101}_{cromosoma(b)}
\end{aligned}
$$

Figura 3.3: Obtención de un nuevo individuo c mediante la cruza de dos individuos padres a y b.

Es posible generar una segunda representación con las subsecuencias que no fueron utilizadas para la representación del nuevo individuo c, la subsecuencia de $(p_c + 1)$ a 19 del padre a y la subsecuencia de 0 a p_c del padre b.

3.1.2. Selección

El operador de selección, como se mencionó al inicio de la Sección 3.1 se utiliza para seleccionar a los mejores individuos de una población para aparearse. De los algoritmos de operadores de selección disponibles [Randy L. Haupt04], se decidió utilizar el algoritmo llamado "selector universal estocástico", dadas sus propiedades estadísticas [Baker87]. Además, este algoritmo es más simple de implementar que el resto de los algoritmos de selección. Éste consiste en generar una tabla con las probabilidades de reproducción de cada uno de los individuos, al igual que en el algoritmo de selección de ruleta; después se calcula un número F_0 al azar dentro del rango de $(0, 1/N)$, donde N es el número de individuos a seleccionar. Para realizar la selección se repite un ciclo de la manera siguiente: se comienza con un $F_c = F_0$ después se selecciona el individuo cuya probabilidad acumulada (la suma de las probabilidades de todos los individuos anteriores al i-ésimo individuo de la tabla) sea mayor que F_c; después de seleccionar al individuo F_c se aumenta en $1/N$, el procedimiento se repite N veces. El Algoritmo 1 describe este procedimiento.

En donde la función **generar_tabla_probabilidad** genera una tabla con la probabilidad de selección de cada uno de los indiviudos de la población. La probabilidad es calculada en base al valor de aptitud de cada individuo; para una población con M indivi-

Algoritmo 1 seleccion_universal_estocastica(poblacion)

```
1: seleccion ← arreglo()
2: tabla ← generar_tabla_probabilidad(poblacion)
3: F0 ← random() * 1/N
4: Fc ← F0
5: for i ← 1 ... N do
6:    indice ← seleccionar_indice_individuo(tabla, F_c)
7:    push(indice, seleccion)
8:    Fc ← Fc + 1/N
9: end for
10: return seleccion
```

duos, si el individuo a_j tiene un valor de aptitud C_j entonces la probabilidad de selección P_j del indivudo será

$$P_j = \frac{C_j}{\sum_{k=1}^{M} C_k} \tag{3.1}$$

La función **seleccionar_indice_individuo** la cual recibe como parámetros la tabla de probabilidad de selección y el valor F_c, regresa el índice de la tabla de probabilidad para el cúal la probabilidad acumulada es mayor que el valor de F_c, i.e. si para el índice k de la tabla de probabilidad se cumple

$$\sum_{i=1}^{k} P_i > F\_c \tag{3.2}$$

donde P_i es la probabilidad de selección del i-ésimo individuo, entonces la función **seleccionar_indice_individuo** regresará el valor de k.

3.1.3. Mutación

Para realizar una mutación primero se seleccionan un número de individuos al azar dependiendo de la probabilidad de mutación. La probabilidad de mutación es manejada en el

rango de 0 a 1 y es recomendable mantenerla baja con un valor de 0.01 [Goldberg00]. Después de que son seleccionados los individuos para mutar, en cada uno de ellos se selecciona la posición de uno de los caracteres del cromosoma aleatoriamente. El carácter en la posición seleccionada será cambiado de 0 a 1, o de 1 a 0. La aplicación de este operador genético se ilustra en la Figura 3.4

$$0101010111101001010 0 \Rightarrow 0101\mathbf{1}101111010010100$$

Figura 3.4: Mutación de un individuo en la posición número 5.

3.1.4. Algoritmo genético con especies

En muchos de los problemas de optimización, no existe un sólo óptimo en la región de búsqueda; es posible que exista más de un óptimo global o varios óptimos locales. A los problemas donde existe más de un óptimo se les llama problemas multimodales. Se han desarrollado varios tipos de algoritmos genéticos para utilizarse en problemas multimodales, estos difieren de los algoritmos genéticos estándar en que tratan de evitar la convergencia de los individuos de la población a un solo óptimo, lo cual se logra generalmente al agrupar los indiviudos en subpoblaciones [Mahfoud95]. El algoritmo utilizado en la presente tesis es el algoritmo genético creado por J-P Li et al. [Li J.-P.02], el cual está basado en el concepto de especies.

El concepto de especies está basado en la observación de que en la naturaleza diferentes especies conviven en un mismo entorno. El algoritmo consiste en dividir una población en subconjuntos llamados especies. Para lograr ésto se selecciona el mejor individuo de toda la población, y todos los individuos con una distancia al individuo seleccionado menor al valor de un parámetro llamado radio, se considera dentro de la misma especie. No hay un método general para determinar el valor para el parámetro del radio, debe calcularse por medio de experimentación u otros métodos [Li J.-P.02]. Este proceso se continua con los individuos que no han sido considerados dentro de una especie. Se selecciona al mejor individuo que no ha sido considerado dentro de una especie y se genera una nueva especie con los individuos que tengan una distancia al individuo menor al radio. A los individuos a partir de los cuales se generan las especies son llamados individuos dominantes o semilla de la especie. Los individuos dominantes seleccionados son reinsertados en la población en

la siguiente generación. El algoritmo de especies se describe en el Algoritmo 2.

Algoritmo 2 algoritmo_genetico_especies(poblacion, funcion_aptitud, generaciones)

```
1: evaluar_poblacion(poblacion, funcion_aptitud)
2: for i ← 1 ... generaciones do
3:    semillas ← seleccionar_individuos_dominantes(poblacion)
4:    seleccionar_individuos(poblacion)
5:    realizar_cruza(poblacion)
6:    evaluar_poblacion(poblacion, funcion_aptitud)
7:    reinsertar_individuos_dominantes(poblacion, semillas)
8: end for
9: return poblacion
```

El algoritmo para la selección de individuos con valores máximos en las subpoblaciones es el utilizado por J-P Li et al. [Li J.-P.02]. Éste consiste en seleccionar el individuo con la aptitud máxima de la población y enseguida seleccionar a todos los individuos con valor de distancia al individuo seleccionado menor o igual al radio. Con los individuos seleccionados se crea una subpoblación y se marcan como seleccionados, el procedimiento se repite con el resto de los individuos no seleccionados, como se ilustra en el Algoritmo 3, en donde X_s es el conjunto de individuos dominantes y $d(x^*, x)$ es la distancia euclidiana [Li J.-P.02].

Como puede observarse en el Algoritmo 2 los individuos dominantes son reinsertados en la población después de aplicar los operadores de selección y cruza, no se aplica el operador de mutación debido a que podría modificar a uno de los individuos dominantes, teniendo como consecuencia la perdida de un posible óptimo local [Li J.-P.02]. Para reinsertar los individuos se sigue también el mismo algoritmo de J-P Li et al. [Li J.-P.02], el cual consiste en reinsertar los individuos dominantes en la población, preferentemente en la misma subpoblación de donde fueron seleccionados, bajo la restricción de que el individuos a reinsertar en una subpoblación debe tener una aptitud mayor que el individuo con el valor de aptitud mínimo dentro de la subpoblación. De no darse el caso, el individuo a

Algoritmo 3 seleccionar_individuos_dominantes(poblacion, r)

1: $X_s \leftarrow \emptyset$
2: **while** hay individuos sin seleccionar **do**
3: buscar al individuo x^* sin seleccionar con la mayor aptitud
4: marcar x^* como seleccionado
5: encontrado $\leftarrow$ falso
6: **for all** $x \in X_s$ **do**
7: **if** $d(x^*, x) \leq r$ **then**
8: encontrado $\leftarrow$ verdadero
9: **end if**
10: **end for**
11: **if** no encontrado **then**
12: $X_s \leftarrow X_s \cup \{x^*\}$
13: **end if**
14: **end while**
15: **return** X_s

reinsertar sustituye al individuo con el valor de aptitud mínimo de la población en general. Un bosquejo del algoritmo se muestra en el Algoritmo 4.

Algoritmo 4 reinsertar_individuos_dominantes(X_s, poblacion)

```
1:  for all x* ∈ X_s do
2:     seleccionar el peor individuo x de la subpoblación de x*
3:     if existe x then
4:        if x es peor que x* then
5:           reemplazar x con x*
6:        end if
7:     else
8:        seleccionar el peor individuo x de la población sin procesar
9:        reemplazar x con x*
10:    end if
11:    marcar x como procesado
12: end for
13: return true
```

El algoritmo de conservación de especies preserva los individuos dominantes en cada generación. Ésto proporciona un mecanismo para encontrar más de un posible óptimo de una función dada. Aunque el desempeño del algoritmo de conservación de especies varia dependiendo del valor asignado al parámetro del radio, es necesario determinar un valor óptimo para el parámetro del radio.

3.1.5. Criterio de terminación para el ciclo del algoritmo

No existe un criterio general para determinar cuando detener un algoritmo genético orientado a la solución de problemas multimodales [Coello99]. Más aún, en el trabajo de Aggarwal [Aggarwal00] no se da un criterio de terminación de forma explícita y en J-P Li et al. [Li J.-P.02] sólo se da un criterio para determinar si un individuo es candidato para máximo global. El criterio de terminación para algoritmo genético utilizado en la presente tesis se describe en el Capítulo 5.

3.2. Optimización de enjambre de partículas

Otro de los algoritmos de optimización inspirados en la naturaleza es el llamado optimización de enjambre de partículas. Este algoritmo está basado en la observación del comportamiento social de los individuos en una especie; por ejemplo los cardúmenes de peces y las parvadas de aves. Cuando un individuo encuentra una región con alimento los demás lo seguirán.

Este algoritmo comparte muchas de las características de los algoritmos genéticos: también comienza con una población inicial, cada partícula también representa un punto en el espacio de búsqueda, y de igual manera cada partícula posee un valor determinado por la función que se quiere optimizar; este valor es utilizado para dirigir el movimiento de las partículas y este representa la cantidad de alimento en la posición donde se encuentra la partícula. Una diferencia con algoritmos genéticos es que no se descartan los individuos de la población inicial, éstos solo actualizan su posición, además no se aplican operadores de cruza o mutación. Cada partícula conserva dos elementos de información: el primero corresponde a la posición de la partícula con el valor máximo (componente social); el segundo corresponde a la posición donde la partícula ha obtenido el mayor valor(componente cognitivo). El nombre de partículas fue dado por lo autores del algoritmo debido que se usa el concepto de posición y velocidad, así es mejor hablar de la posición y velocidad de una partícula más que de un individuo. Ambos algoritmos, genéticos y de optimización de enjambre de partículas, pueden ser utilizados con representaciones continuas o discretas, aunque los algoritmos genéticos inicialmente fueron desarrollados para representaciones discretas [Goldberg00] y los algoritmos de optimización de enjambre de partículas se desarrollaron para representaciones continuas [Kennedy01].

De la misma forma que los algoritmos genéticos, en la optimización de enjambre de partículas se lleva a cabo un proceso iterativo para encontrar el óptimo de una función determinada. Se inicia con un enjambre inicial, una colección de partículas colocadas aleatoriamente dentro del espacio de búsqueda. Una vez generado el enjambre inicial cada partícula es evaluada; la partícula con el valor máximo (o mínimo según sea el problema) es seleccionada y la posición de esta partícula es registrada en cada una de las partículas. Después de la evaluación se calcula una velocidad considerando los dos componentes de información guardados en la partícula. Se calcula una velocidad hacia el máximo global y otra hacia la posición del mejor valor registrado.

En cada iteración se selecciona la partícula con mejor valor, así la posición a donde se dirigen la partículas no siempre es la misma. De la misma forma en cada iteración se actualiza la posición del mejor valor registrado por la partícula; de esta forma los dos componentes de información de la partícula siempre están cambiando.

3.2.1. Cálculo de la velocidad de las partículas

Una vez generado, inicializado y evaluado el enjambre inicial, la velocidad y posición de todas las partículas es actualizada de acuerdo a las reglas dadas en las Ecuaciones (3.3) y (3.4) [Clerc02].

$$v_{new} = \chi[v + C_1 R_1(P_{best} - P) + C_2 R_2(P_{global} - P)] \quad (3.3)$$

$$P_{new} = P + V_{new} \quad (3.4)$$

donde v y P son la velocidad y posición actual, respectivamente; R_1 y R_2 son números aleatorios generados en el rango de $[0, 1]$; C_1 y C_2 son los factores de aprendizaje, los cuales son constantes arbitrarias, pero se recomienda que estas tomen valores mayores a 2 [Kennedy01, Clerc02]; P_{best} es la posición de la partícula con mejor valor de aptitud registrada hasta la iteración actual; P_{global} es la posición de la partícula con el mejor valor de aptitud en el enjambre, en este caso de la especie en la iteración actual; v_{new} y P_{new} son la nueva posición y velocidad de la partícula; la constante χ es calculada de acuerdo a las Ecuaciones (3.5) y (3.6) [Clerc02].

$$\phi = C_1 + C_2 \quad (3.5)$$

$$\chi = \frac{2}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|} \quad (3.6)$$

La constante χ es llamada coeficiente de constricción [Kennedy01, Clerc02, Li06] y evita que las partículas exploren demasiado lejos del espacio de búsqueda. Así no necesitamos una constante v_{max} para limitar la velocidad de las partículas [Li06]. El Algoritmo 5 describe el algoritmo de optimización de enjambre de partículas estándar.

En el Algoritmo 5 la función **generar_enjambre** genera un enjambre inicial con puntos calculados aleatoriamente dentro de los rangos del espacio de búsqueda contenidos en la variable "rangos". La función **calcular_maximo_global** determina la partícula con el

Algoritmo 5 enjambre_particulas(rangos, iteraciones)

```
1: enjambre ← generar_enjambre(rangos)
2: evaluacion_inicial(enjambre)
3: for i ← 1... iteraciones do
4:    maximo_global ← calcular_maximo_global(enjambre)
5:    calcular_velocidades(enjambre, maximo_global)
6:    actualizar_particulas(enjambre)
7: end for
8: return enjambre
```

valor de aptitud máximo dentro de un enjambre, esto en el caso de que se búsque el máximo de una función, debe cambiarse por una función que determine la partícula con el valor de aptitud mínima en caso de minimización. La función **evaluacion_inicial** evalua el valor de aptitud de cada punto en el enjambre en su posición inicial, también inicializa la velocidad, que en este caso se inicializa a cero, y registra la posición y valor de aptitud iniciales como la mejor obtenida hasta el momento.

La función **calcular_velocidades** utiliza las Ecuaciones (3.3) y (3.4) para calcular la nueva velocidad de las partículas en un enjambre y actualizar su posición. La función **actualizar_particulas** determina si la nueva posición de una partícula tiene un valor de aptitud mejor (mayor que en caso de maximizar, menor que para minimizar) que la registrada por la partícula hasta la iteración actual. Si la partícula tiene un valor de aptitud mejor, se actualizará el registro de la partícula con el valor de posición y aptitud actuales. El Algoritmo 5 regresa el enjambre en el estado de la última iteración.

3.2.2. Enjambre de partículas con especies

En la presente tesis se han agregado dos pasos al método base de optimización de enjambre de partículas [Parrott06]; después de que el enjambre inicial es creado, la partícula con el mejor valor de aptitud es seleccionada y todas las partículas con una distancia menor que el valor de un parámetro r son seleccionadas como individuos de una especie. El óptimo global de las partículas en la especie es evaluado con la partícula inicialmente seleccionada.

Las partículas en la especie son marcadas como usadas, y la siguiente partícula con el mejor valor de aptitud es seleccionada para crear una nueva especie. Este procedimiento es repetido hasta que todas las partículas estén dentro de una especie (el valor de aptitud de todas las partículas permanece sin alteraciones). Después de que todas las especies han sido formadas sólo un número de partículas debe permanecer en cada especie, solo las n partículas más cercanas a la partícula con el mejor valor de aptitud son conservadas dentro de la especie, las partículas restantes son reinicializadas en posiciones aleatorias. En el Algoritmo 6 muestra el algoritmo de optimización de enjambre partículas con especies.

Algoritmo 6 enjambre_particulas_especies(rangos, iteraciones, r, individuos_especie)

```
1: enjambre ← generar_enjambre(rangos)
2: evaluacion_inicial(enjambre)
3: for i ← 1... iteraciones do
4:    calcular_especies(enjambre, r, individuos_especie)
5:    calcular_velocidades(enjambre)
6:    actualizar_particulas(enjambre)
7:    reiniciar_particulas_rechazadas(enjambre)
8: end for
9: return enjambre
```

El algoritmo de optimización de enjambre de partículas con especies (Algoritmo 6) requiere de un parámetro adicional a los rangos para el espacio de búsqueda y el número de iteraciones a realizar: el radio r. Al tomar partes del algoritmo genético con especies (ver Sección 3.1.4) también require de un valor para el parámetro del radio. La función **generar_enjambre** es la misma que la utilizada en el Algoritmo 5 descrito en la sección anterior, la función **evaluacion_inicial** también es la misma que en el Algoritmo 5.

La función **calcular_especies** realiza un tarea semejante al Algoritmo 3; selecciona la partícula con el mejor valor de aptitud, pero a comparación del Algoritmo 3 no se almacena un copia de la partícula. La función **calcular_especies** después de seleccionar la partícula con el mejor valor de aptitud la registra como máximo global en todas las

partículas que esten a una distancia menor que el valor del radio. El algoritmo considera sólo un determinado de partículas por especie, en esta misma función se cuentan el número de partículas a las que se les ha asignado un mismo máximo global (i.e. estan dentro de una misma especie), después de que el número máximo de partículas admitido individuos_especie es sobrepasado las siguientes partículas son marcadas para reasignar su posición de manera aleatoria.

Las funciones **calcular_velocidades_especies** y **actualizar_particulas** realizan las mismas tareas que en el Algoritmo 5. La función **reiniciar_particulas_rechazadas** asigna una posición aleatoria en el espacio de búsqueda a las partículas marcadas por **calcular_especies** para reasignación, también realiza la evaluación inicial de las partículas reasignadas.

3.3. Conclusiones del capítulo

En este capítulo se han mostrado los principios básicos de los algoritmos evolutivos, así como la definición de los problemas multimodales y como se pueden abordar estos por medio de los algoritmos evolutivos. Se ha mostrado también que estos son sencillos de implementar, presentan pocos parámetros a inicializar (tamaño de población, porcentaje de selección y mutación, parámetro de radio en caso de problemas multimodales) y no requieren que las funciones a ser optimizadas tengan características como la continuidad o la diferenciabilidad. Estas características son importantes ya que permiten examinar problemas donde no es posible aplicar algoritmos estándar.

Capítulo 4

Algoritmo del Explorador para Optimización Multimodal

En algunos problemas de optimización es necesario encontrar no sólo un óptimo local o global, sino todos los óptimos de una función dada. Varios algoritmos se han desarrollado para este propósito. Los algoritmos "multistart" usan una muestra aleatoria de puntos en el espacio de búsqueda y después utilizan algún algoritmo basado en gradiente para alcanzar diferentes óptimos. Los algoritmos de inteligencia artificial, los cuales incluyen a los algoritmos genéticos y los de optimización por enjambre de partículas, también son utilizados para resolver problemas multimodales al agrupar los individuos en "especies o subpoblaciones. Existen muchos métodos para agrupar individuos en subpoblaciones [Mahfoud95], el método más utilizado esta basado en agrupar los individuos por su cercanía en el espacio de búsqueda [Li J.-P.02].

Ambos tipos de métodos, "multistart" y evolutivos, presentan desventajas; los algoritmos "multistart" que usan algoritmos basados en gradiente dependen de la propiedad de diferenciabilidad de la función que se quiere optimizar, y bajo algunas condiciones éstos no convergen. Los algoritmos de inteligencia artificial como los algoritmos genéticos y los de optimización por enjambre de partículas tienen un comportamiento no determinista y dependen de parámetros de agrupamiento que tienen que ser ajustados por medio de experimentación o con métodos auxiliares.

Es de especial interés el método de "especies" de algoritmos evolutivos basado en la agrupación por cercanía, ya que éste ha presentado buenos resultados en problemas

multimodales. El método de agrupación en especies es usado tanto en algoritmos genéticos [Li J.-P.02] como en optimización de enjambre de partículas [Li06]. El método consiste en agrupar todas los individuos (o partículas) cuya distancia a un individuo llamado semilla [Li J.-P.02] sea menor al valor de un parámetro llamado radio. No existe un método general para la estimación del valor del radio y este valor puede afectar el desempeño del algoritmo evolutivo que lo utilice [Li J.-P.02].

En esta tesis se presenta, como una alternativa a los algoritmos evolutivos basados en el método de "especies", un algoritmo que descompone el espacio de búsqueda en regiones, cada una conteniendo un sólo óptimo. El algoritmo está basado en la idea de usar una muestra inicial aleatoria uniforme; la evaluación de la función en cada punto de la muestra forma una representación discreta de la función que se quiere optimizar; después se usa un algoritmo incremental de envoltura convexa [Franco P. Preparata85] para hacer crecer las regiones que contienen óptimos.

El algoritmo está basado en dos suposiciones muy importantes: la primera, siguiendo el teorema de Nyquist [Nyquist28, Shannon49], es que se cuenta con una muestra con una frecuencia suficientemente alta para poder detectar todas las características de la función (i.e. óptimos); segunda, la función a ser optimizada debe mostrar un número finito de óptimos en la región de análisis.

4.1. Optimización multimodal

En muchos de los problemas de optimización es posible que la función objetivo presente más de un óptimo. Los óptimos que presenta un función pueden ser locales o globales. Varios autores han tratado este problema con algoritmos evolutivos [Mahfoud95, Li J.-P.02, Petrowski97, Aggarwal00, Coello99, Randy L. Haupt04, Miller96], aunque todos estos métodos presentan parámetros que para ser ajustados requieren del conocimiento de la distribución de los óptimos de la función; en general no se cuenta con este conocimiento.

4.2. Detección de regiones

La determinación de regiones está basada en el hecho de que un muestreo del espacio de búsqueda puede ser considerado como una representación discreta de la función f analizada. En este trabajo se propone un algoritmo incremental para construir una envoltura

convexa [Franco P. Preparata85] alrededor de cada óptimo en la representación discreta de la función f. Para describir el algoritmo se supondrá que se está buscando un máximo de una función f dada. Para la búsqueda de un mínimo simplemente hay que cambiar en los enunciados mínimo por máximo y viceversa donde sea necesario.

El método propuesto se realiza en los siguientes pasos:

1 Se genera un muestreo aleatorio dentro del espacio de búsqueda y se evalúa.

2 Se selecciona el punto P_s con valor máximo de aptitud, y los puntos de la muestra se ordenan por distancia con respecto a P_s.

3 Se selecciona un subconjunto de los puntos más cercanos a P_s para crear la envoltura convexa inicial; el número de puntos seleccionados incluyendo P_s deben ser suficientes para crear una envoltura convexa. Para un espacio n-dimensional, necesitamos $n + 1$ puntos para crear una envoltura convexa n-dimensional.

4 Después de crear la envoltura convexa inicial, se selecciona el siguiente punto P más cercano a P_s que no es parte de la envoltura convexa. Se determinan las facetas que son visibles por el punto, y se verifica que el valor de f en el punto sea menor que el valor de aptitud de al menos un punto en las facetas visibles; si se cumple la condición, el punto es agregado a la envoltura convexa. Si no se cumple la condición, esto indica que un mínimo ha sido sobrepasado y que P pertenece a otra región.

5 Cada punto que es parte de una envoltura convexa es marcado como utilizado. Si no es posible crear una envoltura convexa inicial (i.e. no hay suficientes puntos sin utilizar) sólo P_s es marcado como utilizado. El método se repite hasta que no queden puntos sin utilizar.

Para ilustrar el funcionamiento del algoritmo utilizaremos la siguiente función vectorial, definida con el siguiente sistema de ecuaciones

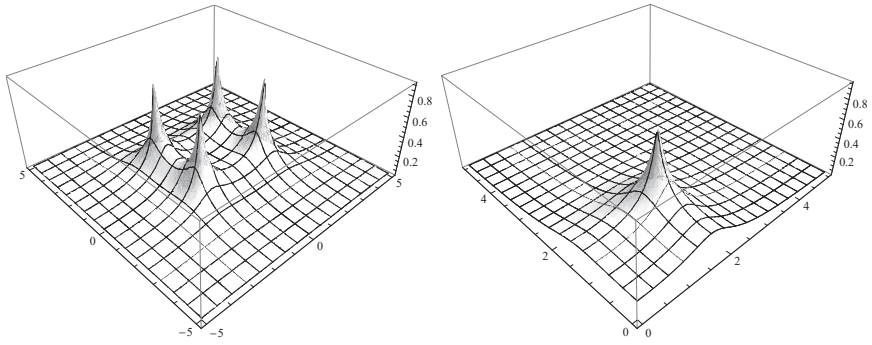
$$\mathbf{f} = \begin{bmatrix} f_1([x_1, x_2]^T) \\ f_2([x_1, x_2]^T) \end{bmatrix} = \begin{bmatrix} x_1^2 + x_2^2 - 4 \\ x_1^2 - x_2^2 - 1 \end{bmatrix} \tag{4.1}$$

Las soluciones de la función $\mathbf{f}([x_1, x_2]^T)$ representan la intersección entre una circunferencia y una hipérbola. Ésto es, el sistema tiene cuatro soluciones, una por cada cua-

drante de $\mathbb{R}^2$. Además es una función con rango y dominio en $\mathbb{R}^2$. Para obtener una función objetivo $g : \mathbb{R}^2 \to \mathbb{R}$, utilizamos la composición

$$g([x_1, x_2]^T) = \frac{1}{1 + |\mathbf{f}([x_1, x_2]^T)|} \tag{4.2}$$

con ésto obtenemos una función con dominio $\mathbb{R}^2$ y rango $\mathbb{R}$, donde los ceros de la función $\mathbf{f}$ son mapeados a uno y $g([x_1, x_2]^T) \to 0$ cuando $\mathbf{f}([x_1, x_2]^T) \to \infty$. La Figura 4.1(a) presenta la gráfica de la función g para los rangos de $x_1 \in [-5, 5]$ y $x_2 \in [-5, 5]$ y la Figura 4.1(b) para $x_1 \in [0, 5]$ y $x_2 \in [0, 5]$; en la segunda región se mostrará como se desarrolla el método propuesto.



(a) (b)

Figura 4.1: Gráfica de la función $g([x_1, x_2]^T)$: a) rangos de $x_1 \in [-5, 5]$ y $x_2 \in [-5, 5]$ y b) cuadrante positivo

Como primer ejemplo se genera una distribución aleatoria de puntos dentro del cuadrante positivo, la Figura 4.2(a) muestra las líneas de nivel de la función g para el cuadrante positivo (ver Figura 4.1(b)) y la muestra aleatoria de puntos.

En la Figura 4.2(b) se muestra la envoltura convexa inicial generado con el punto con el valor máximo de aptitud del muestreo aleatorio y los dos puntos mas cercanos a él. La Figura 4.2(c) muestra una etapa intermedia; la envoltura convexa se ha extendido verificando los puntos más cercanos al punto inicial. Se puede observar de las curvas de nivel de la Figura 4.2(c) y de la Figura 4.1(b) que la envoltura convexa ha crecido en una dirección de descenso. La Figura 4.2(c) muestra la envoltura convexa final. Es claro ver que la envoltura convexa encierra casi en su totalidad la misma región que la curva de nivel con valor más alto. Comparando las Figuras 4.1(b) y 4.2(d) puede observarse que el máximo de

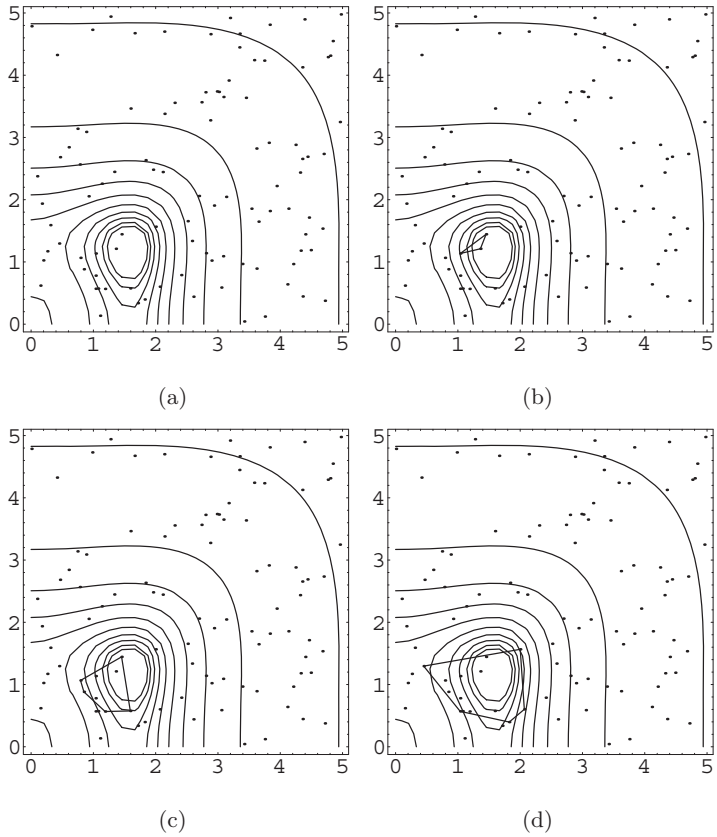


Figura 4.2: Varios estados en la creación de una envoltura convexa: a) puntos de la rejilla, b) envoltura convexa inicial inicial formado por 3 facetas, c) estado intermedio de la envoltura convexa, d) estado final de la envoltura convexa.

la función g está dentro de la envoltura convexa final.

Hay varias observaciones que hacer de este ejemplo, la más importante es acerca del la distribución inicial de los puntos; una distribución aleatoria no es necesariamente uniforme, como puede observarse en la Figura 4.2(a). Ésto tiene como consecuencia que la condición de tener un frecuencia de muestreo suficientemente alta puede no ser cumplida y puede resultar en que algunos máximos de la función que se examina no sean detectados.

La Figura 4.3(a) muestra una distribución inicial uniforme dentro del mismo cuadrante que el ejemplo anterior y para la misma función g. En la Figura 4.3(b) muestra la primera envoltura convexa generada; a diferencia del ejemplo anterior con una distribución aleatoria, comparando con la Figura 4.2(b), puede verse claramente cual es el punto de la

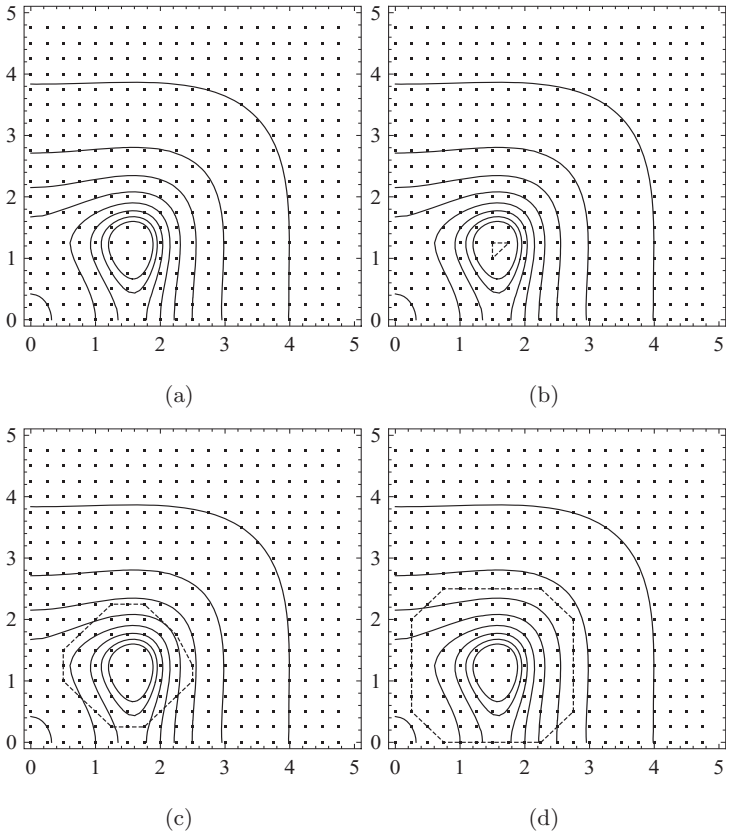


Figura 4.3: Varios estados en la creación de una envoltura convexa: a) puntos de la rejilla, b) envoltura convexa inicial formado por 3 facetas, c) estado intermedio de la envoltura convexa, d) estado final de la envoltura convexa.

envoltura convexa inicial con el valor de aptitud más alto, el punto inicial. En la Figura 4.3(c) se puede observar con claridad que la envoltura convexa ha cubierto por completo la región de la curva de nivel más alta, además de otras regiones con curvas de nivel de valor más bajo en dirección descendente. Además, la envoltura convexa tiene una forma más regular. En la Figura 4.3(d) se muestra la envoltura convexa final; éste ha cubierto por completo no sólo la región con la curva de nivel de valor más alto sino todas las curvas de nivel con valores más bajos de todo el cuadrante. Se puede observar, al comparar las Figuras 4.3(d), 4.1(a) y 4.1(b), que el método se detuvo justo al comenzar un comportamiento ascendente de la función g.

De los ejemplos mostrados puede observarse que el algoritmo tiene un mejor desem-

peño con una muestra inicial uniforme, al cubrirse de forma completa las regiones con valores altos, además se evita el problema de la frecuencia de muestreo irregular de una distribución aleatoria. Sin embargo en el muestreo aleatorio se utilizaron menos puntos; en el primer ejemplo sólo fueron usados 100 puntos en el muestro inicial, que es la cantidad de puntos que se utilizarían para un algoritmo genético en este problema específico. En el segundo se utilizaron 400 puntos para generar una rejilla rectangular uniforme con una separación entre puntos de 0.2 entre cada punto. No obstante, al usar el muestreo aleatorio también fue encontrada una región que contenía el máximo. Ésto debe considerarse sobre todo cuando se tengan problemas en dimensiones más altas, como lo muestra la Sección 4.5.

El Algoritmo 7 muestra el seudocódigo para el método propuesto.

4.3. Filtrado de regiones

El algoritmo ilustrado en la Sección anterior puede producir regiones que no contengan un óptimo local. Es necesario filtrar la regiones para asegurar que dentro de cada una de ellas existirá un óptimo de la función analizada. Dos criterios de filtrado han sido desarrollados hasta el momento: descartar las regiones que no crecen, y realizar un análisis de las regiones cuyo punto P_s inicial es parte de la envoltura convexa que limita la región.

En algunos casos, inmediatamente después de que la envoltura convexa ha sido creada, una condición para terminar su crecimiento es alcanzada. Estas condiciones son: el siguiente punto más cercano ya ha sido utilizado, o el valor de aptitud del siguiente punto más cercano es mayor que el valor de aptitud de todos los puntos que forman parte de las facetas visibles. Ambas condiciones indican que la envoltura convexa inicial está en un valle y no es posible encontrar un óptimo dentro de la envoltura convexa inicial. Las envoltura convexa que no crecen son descartadas; ésto se realiza por comparación del los puntos en la envoltura convexa inicial y los puntos en la envoltura convexa final.

Cuando la envoltura convexa inicial crece pero el punto inicial P_s es parte de la envoltura convexa, éste ha crecido sólo en una dirección y es posible que no exista un óptimo dentro de la envoltura convexa. En este caso se aplica una prueba simple: se calcula el centroide de la envoltura convexa, después se crea un vector del centroide al punto inicial P_s, un punto P_o es calculado fuera de la envoltura convexa en la dirección del vector creado; el nuevo punto P_o está a sólo una pequeña distancia de P_s. Si el valor de aptitud de P_o es menor que el valor de aptitud de P_s la envoltura convexa es preservada, de otra forma es

Algoritmo 7 seleccion_inteligente_regiones(R)

```
1: R[]: arreglo que contiene a los puntos del a rejilla.
2: Rd[]: arreglo para ordenar los puntos de la rejilla por distancia.
3: N: dimensión del espacio de búsqueda
4: CH[]: arreglo de puntos que forman una envoltura convexa
5: LCH[]: arreglo de envolturas convexas
6: F[]: arreglo de puntos que forman facetas
7: R ← ordenar_aptitud_descendente(R[])
8: while Existen puntos sin usar do
9:   P_s ← seleccionar_mejor_punto_disponible(R[])
10:  Rd[] ← ordenar_distancia_ascendente(R[], P_s)
11:  inicial[] ← primeros_n_puntos(Rd[], N)
12:  if no_usado(inicial[]) then
13:    marcar_utilizado(inicial[])
14:    CH[] ← crear_ec_inicial(I_s, inicial[])
15:    TERMINADO ← false
16:    for j = N + 1 to longitud_de(Rd[]) & !TERMINADO do
17:      P ← Rd[j]
18:      if no_usado(P) then
19:        F[] ← facetas_visibles(P, CH[])
20:        FI[] ← puntos_en(F[])
21:        if aptitud(P) < aptitud_maxima(FI[]) then
22:          insertar(P, CH[])
23:          marcar_usado(P)
24:        else
25:          TERMINADO ← true
26:        end if
27:      else
28:        TERMINADO ← true
29:      end if
30:    end for
31:    if aplicar_filtros(CH) then
32:      agregar CH[] a LCH[]
33:    end if
34:  end if
35: end while
36: return LCH[]
```

descartada.

4.4. Completitud y correctitud

Un resultado importante del algoritmo del explorador es que garantiza que todos las envolturas convexas creadas contienen un óptimo dentro de ellos, y que todos los óptimos del dominio dado están encerrados por una envoltura convexa. La primera afirmación provee correctitud, mientras que la segunda provee completitud para el algoritmo.

El resultado después de aplicar el algoritmo en una retícula es un conjunto de regiones, donde cada una de ellas esta delimitada por una envoltura convexa. Cada región contiene un punto con una valor mayor que cualquier otro punto dentro o en la envoltura convexa, como lo indica el siguiente teorema:

Teorema 2 *Si una envoltura convexa CH es creada por el algoritmo del explorador contiene el punto inicial P_m dentro de la envoltura convexa, y la función f es continua, entonces CH contiene un óptimo de la función objetivo f.*

Demostración Dado el procedimiento de construcción de la envoltura convexa por el Algoritmo 7, la envoltura convexa es un conjunto cerrado en R^n. Siguiendo el teorema del valor extremo, aplicado a una función continua f, el teorema implica que f tiene un máximo y un mínimo en la envoltura convexa. Si el punto inicial está dentro de la envoltura convexa, i.e., no pertenece al conjunto de puntos que conforman la envoltura convexa; el máximo de la función f está dentro de la envoltura convexa.

Teorema 3 *Toda envoltura convexa CH producido por el Algoritmo 7, que pase el proceso de filtrado, contiene un óptimo de la función f.*

Demostración Tenemos dos casos: el punto inicial P_m se encuentra dentro de la envoltura convexa. Este caso es demostrado por el Teorema 2. En otro caso el punto inicial P_m forma parte de la envoltura convexa, el proceso de filtrado descrito en la Sección 4.3 asegura que P_m es un máximo local de la función f. Los filtros descartan las envolturas convexas que no contienen un máximo local de la función f.

Si las condiciones mencionadas en el inicio del presente capítulo son cumplidas, cualquier óptimo en la región examinada será encerrado dentro de una envoltura convexa, como es mencionado en el siguiente teorema:

Teorema 4 *Después de aplicar el Algoritmo 7, cada óptimo de una función f objetivo dada con dominio $X \subset R^n$, está encerrado en una envoltura convexa.*

Demostración Suponemos que dos máximos a y a' de la función f forman parte de una envoltura convexa creado por el Algoritmo 7; si tenemos una frecuencia de muestreo suficientemente alta, existe un punto b entre a y a'. Como a y a' son máximos de f, $f(b)$ tiene un valor menor que $f(a)$ y $f(a')$. Suponga que el algoritmo comienza con a como primer punto de la envoltura convexa; entonces el Algoritmo 7 debe utilizar primero b para probar si éste es agregado a la envoltura convexa, después prueba a', debido a que $f(a')$ tiene un valor mayor que $f(b)$; el algoritmo se detiene y a' no es incluido en la envoltura convexa. El mismo razonamiento se realiza si el algoritmo comienza con a'. Lo anterior implica que no pueden existir dos máximos en la misma envoltura convexa.

4.5. Complejidad

Un análisis preliminar muestra que el Algoritmo 7 tiene en el peor caso una complejidad de $O(n^2 \log n)$, donde n es el número de puntos en el muestreo. Ésto se debe a que el Algoritmo 7 contiene al algoritmo incremental de envoltura convexa dentro de un ciclo que recorre todos los puntos del muestreo inicial. La complejidad del algoritmo incremental de envoltura convexa es de orden $O(n \log n)$ [Franco P. Preparata85, O'Rourke98]. Ésta es la complejidad con respecto al número de puntos del muestreo inicial.

Para el caso de un muestreo uniforme, si N es la dimensión del problema, tenemos $n = O(k^N)$ donde k es constante, la complejidad del algoritmo en términos de la dimensión del problema es $O(k^{2N} N \log k)$. Simplificando tenemos que la complejidad es de orden $O(N2^N)$. Ésto es, para una N dada, el algoritmo es polinomial en k, y por lo tanto computacionalmente eficiente, pero la complejidad crece exponencialmente con respecto a N.

En el caso de los filtros, la comparación de la envoltura convexa inicial con el final es realizada en $O(N)$. En el caso donde el punto inicial forma parte de la envoltura convexa inicial, las coordenadas del centroide son la media aritmética de los valores de las coordenadas de los puntos que conforman la envoltura convexa; así el centroide es calculado en $O(F)$, siendo F el número de facetas en la envoltura convexa.

4.6. El algoritmo n-dimensional de envoltura convexa

La mayoría de las implementaciones disponibles del algoritmo incremental de envoltura convexa sólo muestran ejemplos en dos y tres dimensiones, y sugieren que pueden ser extendidos a dimensiones más altas. Un problema con estos algoritmos es la representación de las facetas y los bordes en dimensiones más altas. La mayoría de ellos usan diagramas de Voronoi para su representación. Esta aproximación requiere de una gran cantidad de memoria y un gran número de operaciones para preservar la estructura de las facetas y bordes. Para simplificar el algoritmo, tanto en uso de memoria como en operaciones, se representa cada faceta con sólo el conjunto de puntos que la forman, con un orden adecuado, y se define un algoritmo para obtener los bordes de una faceta por medio de una función recursiva.

4.6.1. Representación de una faceta

Una operación importante es la selección de las facetas que son visibles por un punto nuevo. Para lograr ésto, es necesario calcular el determinante formado con las coordenadas de todos los puntos en la faceta y el punto que se esta comparando. Cada punto representa una columna del determinante; al determinante también se le agrega un renglón conformado con unos al final. Con ésto se logra un determinante de $n \times n$ para un espacio de n dimensiones. Como un ejemplo en tres dimensiones, considere una faceta formada por los puntos p_1, p_2, p_3, representada por las líneas (p_1, p_2), (p_2, p_3) y (p_3, p_1) (ver Figura 4.4). Al compararla con el punto p_4 debemos calcular el determinante

$$\mathrm{Det}(p_1 p_2 p_3 p_4) \tag{4.3}$$

O, de forma explícita

$$\mathrm{Det}(p_1 p_2 p_3 p_4) = \begin{vmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \\ 1 & 1 & 1 & 1 \end{vmatrix} \tag{4.4}$$

Es de notarse el orden de los puntos; la faceta puede representarse por los puntos en el orden en el cual aparecen en el determinante, i.e., la representación de la faceta en

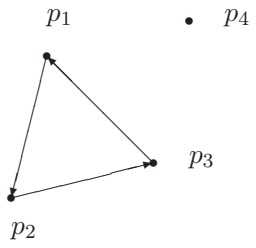


Figura 4.4: Una faceta en tres dimensiones.

la Figura 4.4 será $p_1p_2p_3$. Los bordes de la faceta son representados en la misma forma, entonces, los bordes serán p_1p_2, p_2p_3 y p_3p_1. Para construir una faceta a partir de un borde en nuestra representación es simple; siguiendo el mismo ejemplo en un espacio de tres dimensiones, si queremos construir una faceta a partir del borde (p_3p_1) (ver Figura 4.5) y el punto p_n, necesitamos dos bordes mas, los bordes (p_1p_n) y (p_np_3); en los términos del determinante de la Ecuación (4.3), es el equivalente a agregar el punto p_n al final de la lista de los puntos del borde, i.e., si queremos saber si nuestra nueva faceta formada con los bordes (p_3p_1), (p_1p_n) y (p_np_3) es visible por un punto p_4 necesitamos calcular

$$\mathrm{Det}(p_3p_1p_np_4) \tag{4.5}$$

esto es, para construir una nueva faceta con el borde (p_3p_1) y el punto p_n, en nuestra representación, es el equivalente a comenzar con el borde p_3p_1 y agregar p_n al final, la nueva faceta es $p_3p_1p_n$. Para calcular de forma adecuada el determinante , el orden de los puntos debe ser preservado, a partir de este hecho podemos generalizar la construcción de una nueva faceta, si tenemos un borde n-dimensional (en nuestra representación) $p_1p_2 \ldots p_{n-1}$, y e punto p_n, la faceta resultante construida con el bordo y el punto es $p_1p_2 \ldots p_{n-1}p_n$. A partir del método de construir una faceta, podemos afirmar que, si tenemos una faceta $p_1p_2 \ldots p_{n-1}p_n$, en un espacio n-dimensional, uno de sus bordes es $p_1p_2 \ldots p_{n-1}$.

Si sólo nuestra faceta $p_1p_2p_3$ es visible por el punto p_4 y queremos construir nuevas facetas a partir de los bordes de $p_1p_2p_3$, en este caso de tres dimensiones, sólo necesitamos construir una nueva faceta a partir de cada uno de los bordes, en la representación mencionada anteriormente las facetas resultantes serán $p_1p_2p_4$, $p_2p_3p_4$ y $p_3p_1p_4$.

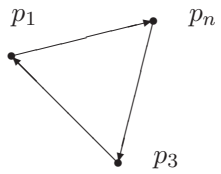


Figura 4.5: Una nueva faceta creada con el borde p_3p_1 y el punto p_n. La nueva faceta es representada como $p_3p_1p_n$.

4.6.2. Construcción de la envoltura convexa inicial en tres y cuatro dimensiones

Si queremos construir una envoltura convexa inicial en tres dimensiones, después de verificar que el signo del determinante de la Ecuación (4.3) es positivo, sólo necesitamos "invertir" la dirección de cada uno de los bordes de $p_1p_2p_3$ y seguir las indicaciones de construcción de una faceta a partir de un borde tridimensional mencionado en la sección anterior. Los bordes de $p_1p_2p_3$ son p_1p_2, p_2p_3 y p_3p_1, los bordes invertidos son p_2p_1, p_3p_2 y p_1p_3. La nueva faceta construida a partir del borde invertido p_1p_3 y el punto p_4 es $p_1p_3p_4$ (ver Figura 4.6). El resto de las facetas son construidas en la misma forma; la envoltura convexa inicial estará entonces formado con las facetas: $p_1p_2p_3$, $p_2p_1p_4$, $p_3p_2p_4$ y $p_1p_3p_4$.

Debe notarse también que, al invertir el orden de los dos primeros puntos de la representación de un borde, cambia el signo del determinante en la Ecuación (4.3), y del método de construcción de una faceta a partir de un borde (sólo agregamos un punto a la representación del borde), es el equivalente a invertir la faceta completa. Si tenemos la faceta $p_1p_2p_3$ la faceta invertida es $p_2p_1p_3$ (véase la Figura 4.7).

En el caso de cuatro dimensiones después de verificar el valor y signo del determinante $\mathrm{Det}(p_1p_2p_3p_4p_5)$ las facetas se pueden determinar de la siguiente manera: una de las facetas de $p_1p_2p_3p_4p_5$ (como se mostró en la Sección 4.6.1) debe ser $p_1p_2p_3p_4$ y las facetas restantes pueden construirse a partir de los bordes de $p_1p_2p_3p_4$; una faceta en cuatro dimensiones es una envoltura convexa mínimo en tres dimensiones, i.e., una faceta en cuatro dimensiones es un tetraedro formado por cuatro triángulos. Del ejemplo en tres dimensiones sabemos que los bordes de $p_1p_2p_3p_4$ son $p_1p_2p_3$, $p_2p_1p_4$, $p_3p_2p_4$ y $p_1p_3p_4$, sólo necesitamos "invertir" cada uno de los bordes y construir las facetas restantes con p_5. Los bordes son

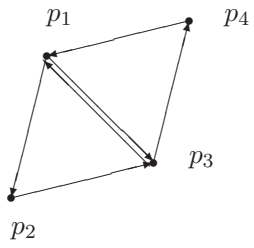


Figura 4.6: La faceta inicial representada por $p_1p_2p_3$ y la segunda faceta creada invirtiendo el borde p_3p_1 y el punto $p4$. La segunda faceta es representada por $p_1p_3p_4$.

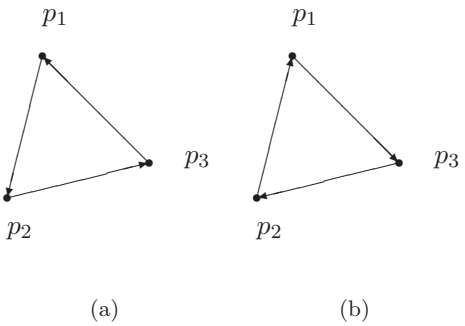


Figura 4.7: a) faceta representada por $p_1p_2p_3$, b) faceta invertida representada por $p_2p_1p_3$.

invertidos simplemente invirtiendo el orden de los dos primeros puntos; entonces las facetas de la envoltura convexa inicial en cuatro dimensiones son: $p_1p_2p_3p_4$, $p_2p_1p_3p_5$, $p_1p_2p_4p_5$, $p_2p_3p_4p_5$ y $p_3p_1p_4p_5$.

Es posible construir de manera fácil por medio de recursión, y así los bordes de una faceta en cualquier dimensión. Una faceta n-dimensional es una envoltura convexa inicial en dimensión $n-1$, entonces podemos utilizar una representación para una faceta utilizando sólo la colección de puntos que la forman en el orden en el que aparecen en el determinante utilizado para determinar la visibilidad de la misma desde un punto, y podemos extraer la información de los bordes de una faceta usando un método recursivo, con $n-2$ pasos recursivos. El Algoritmo 8 muestra la función recursiva para construir los bordes de una faceta.

Algoritmo 8 bordes_faceta(faceta)

```
1: resultado ← {}
2: longitud ← longitud_faceta(faceta)
3: if longitud == 3 then
4:    resultado ← append(resultado, [faceta[0],faceta[1]])
5:    resultado ← append(resultado, [faceta[1],faceta[2]])
6:    resultado ← append(resultado, [faceta[2],faceta[0]])
7: else
8:    bordes ← bordes_faceta(sub_array(faceta, 0, longitud - 2))
9:    for all borde in bordes do
10:      borde ← invertir_borde(borde)
11:      borde ← append(borde, faceta[longitud - 1])
12:      resultado ← append(ridge, result)
13:   end for
14:   resultado ← append(sub_array(faceta, 0, longitud - 2), resultado)
15: end if
16: return resultado
```

4.6.3. Adición de un punto a una envoltura convexa n-dimensional

Con la representación descrita en la Sección 4.6.1 para facetas n-dimensionales, es fácil construir el determinante para determinar si una faceta es visible por un punto dado. Pero no podemos evitar el calculo de los bordes para cada conjunto de facetas, el conjunto de facetas visibles y el conjunto de las facetas no-visibles. Para reducir el número de facetas utilizados usadas en los cálculos, la implementación del algoritmo usa las operaciones de conjuntos, en particular la intersección de conjuntos. Además, se definen dos relaciones para ayudar a reducir el número de bordes que necesitamos calcular.

Definición 1 *Dos bordes a y b en un espacio n-dimensional son contiguos si estos tienen los mismo puntos.*

Definición 2 *Dos facetas A y B en un espacio n-dimensional son contiguas si comparten $n-1$ puntos.*

Para reducir el número de bordes a ser calculados, primero reducimos los conjuntos de facetas visibles y facetas no-visibles, determinando cuales facetas tienen bordes en el horizonte, i.e., la intersección del conjunto de facetas visibles y el conjunto de facetas no visibles usando la relación en la Definición 2.

Una vez que se han determinado las facetas que tienen bordes en el horizonte, calculamos los bordes de las facetas de ambos conjuntos, así tenemos el conjunto de los bordes de las facetas visibles R_v, y el conjunto de los bordes de las facetas no visibles R_n. Para determinar los bordes de los cuales construiremos las nuevas facetas, calculamos la intersección de los conjuntos R_v y R_n. Los bordes que necesitamos son aquellos en la intersección que son elementos del conjunto R_n.

Construir las nuevas facetas es simple: sólo necesitamos agregar el punto que esta siendo comparado al final de la representación de cada borde, de manera semejante a como se construyo la envoltura convexa inicial para el caso de cuatro dimensiones. Entonces la nueva envoltura convexa es la unión del conjunto de facetas no visibles y el conjunto de las nuevas facetas creadas.

4.6.4. Implementación del algoritmo incremental de envoltura convexa para n dimensiones

En esta sección se muestra el seudocódigo para la implementación del algoritmo incremental de envoltura convexa. El algoritmo esta basado en el número de coordenadas de los puntos, así el mismo algoritmo puede ser usado para cualquier dimensión. También se muestran figuras del resultado del algoritmo, así como de las etapas intermedias de calculo.

El Algoritmo 9 ilustra la implementación del algoritmo incremental de envoltura convexa. Este algoritmo sólo requiere algunas pocas funciones de soporte; la función **seleccionar_puntos_iniciales** debe regresar como resultado los primeros $n+1$ puntos tales que el determinante formado por las coordenadas de los puntos del resultado, sea positivo. Las funciones **calcular_facetas_visibles** y **calcular_facetas_no_visibles** utilizan el método estándar para determinar si una faceta es visible por un punto (calcular el valor del determinante de la Ecuación (4.3)). La envoltura convexa inicial es calculada al considerarla como una faceta en dimensión $n+1$ y calcular sus bordes. El ciclo principal del Algoritmo 9 es descrito con la ayuda de la Figura 4.8.

En la Figura 4.8(a) se muestra una envoltura convexa y un punto a ser agregado. Primero se calculan los conjuntos de facetas visibles y no visibles, en el caso de que el punto sea interior, el conjunto de facetas visibles es el conjunto vacío. La Figura 4.8(b) muestra un conjunto de facetas visibles por el punto dado, y la Figura 4.8(c) muestra el conjunto que corresponde a las facetas no visibles. Ambos conjuntos son calculados en los renglones 14 y 15 del Algoritmo 9; lo cual se realiza calculando el determinante mencionado anteriormente para cada faceta de la envoltura convexa. Los conjuntos pueden ser calculados por una misma función, en el Algoritmo 9 es realizado en dos pasos para propósitos de claridad.

Para reducir el número de bordes a calcular, las facetas que no tienen bordes en el horizonte (los bordes en el límite entre las facetas visibles y no visibles) son eliminadas de ambos conjuntos de facetas, visibles y no visibles. Esto se realiza calculado la intersección de los conjuntos de facetas visibles y no visibles considerando la igualdad entre elementos la relación de facetas contiguas definida anteriormente. Las implementaciones estándar de la función de intersección, no garantizan que los elementos en el resultado pertenezcan al conjunto que se pasa como primer argumento. Para el correcto funcionamiento del Algoritmo 9 es necesario garantizar que los elementos en el resultado de la función de intersección pertenezcan al conjunto que se pasa como primer argumento, para cumplir con este requisito

Algoritmo 9 envoltura_convexa_incremental(puntos)

```
1: facetas_visibles ← arreglo de facetas que son visibles por un punto dado.
2: facetas_no_visibles ← arreglo de facetas que no son visibles por un punto dado.
3: horizonte_visible ← arreglo de facetas que son visibles por un punto dado y
   que contienen bordes en el horizonte.
4: horizonte_no_visible ← arreglo de facetas que no son visibles por un punto dado
   y que contienen bordes en el horizonte.
5: bordes_visibles ← arrego de los bordes de las facetas en horizonte_visible.
6: bordes_no_visibles ← arreglo de los bordes de las facetas en horizon-
   te_no_visible.
7: horizonte ← arreglo de bordes en el horizonte.
8: facetas_nuevas ← arreglo de facetas calculadas de los bordes en el horizonte y
   un punto dado.
9: numero_de_puntos ← longitud(puntos)
10: dimension ← longitud(puntos[0])
11: puntos_iniciales ← seleccionar_puntos_iniciales(puntos, dimension + 1)
12: envoltura_convexa ← bordes_faceta(puntos_iniciales)
13: for i ← (dimension + 1) to numero_de_puntos do
14:     facetas_visibles ← calcular_facetas_visibles(envoltura_convexa, puntos[i])
15:     facetas_no_visibles ← calcular_facetas_no_visibles(envoltura_convexa,
        puntos[i])
16:     horizonte_visible ← interseccion(facetas_visibles, facetas_no_visibles, face-
        ta_contigua)
17:     horizonte_no_visible ← interseccion(facetas_no_visibles, facetas_visibles, fa-
        ceta_contigua)
18:     bordes_visibles ← calcular_bordes(horizonte_visible)
19:     bordes_no_visibles ← calcular_bordes(horizonte_no_visible)
20:     horizonte ← interseccion(bordes_visibles, bordes_no_visibles, bor-
        de_contiguo)
21:     facetas_nuevas ← calcular_facetas(horizonte, puntos[i])
22:     envoltura_convexa ← append(fecetas_no_visibles, facetas_nuevas)
23: end for
24: return envoltura_convexa
```

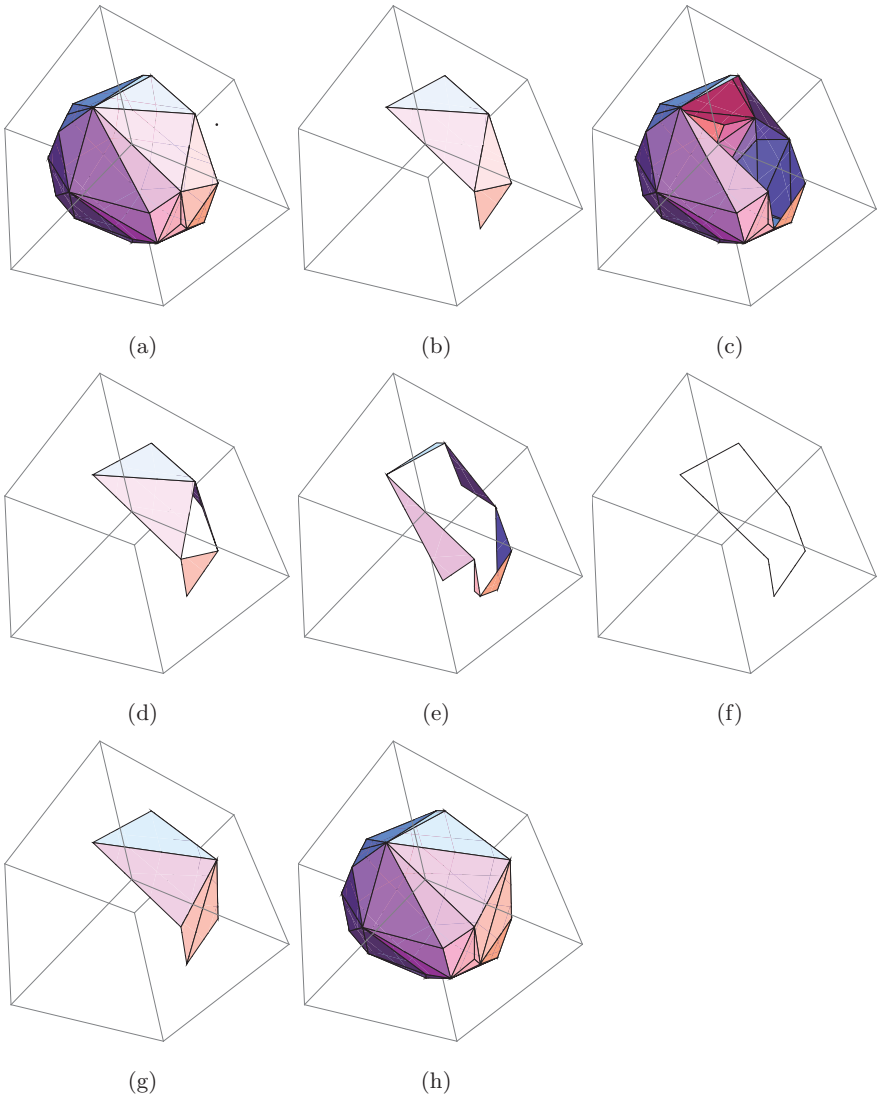


Figura 4.8: Pasos para agregar un punto a la envoltura convexa. a) envoltura convexa actual y el punto que será agregado, b) facetas visibles por el punto a agregar, c) facetas no visibles por el punto a agregar, d) facetas visibles con bordes en el horizonte, e) facetas no visibles con bordes en el horizonte, f) horizonte calculado, g) facetas nuevas calculadas con el horizonte y el punto a agregar, h) envoltura convexa nueva.

se implenta una función de intersección. La función de intersección implementada regresa como resultado los elementos del primer conjunto que forman parte de la intersección con el segundo conjunto, así las facetas visibles que tienen bordes en el horizonte son calculadas al usar como primer argumento el conjunto de facetas visibles y como segundo argumento el conjunto de facetas no visibles; de manera semejante se calculan las facetas no visibles con bordes en el horizonte. En las líneas 16 y 17 del Algoritmo 9 se realizan estos cálculos y los resultados son mostrados en la Figura 4.8(d) y (e) para las facetas visibles y no visibles respectivamente.

Después son calculados los bordes sólo para las facetas que tienen bordes en el horizonte; ésto se realiza en las líneas 18 y 19 del Algoritmo 9. La intersección de los conjuntos de bordes da como resultado los bordes que forman el horizonte, éste es calculado en la línea 20 del Algoritmo 9 y el resultado es mostrado en la Figura 4.8(f). Entonces son calculadas nuevas facetas utilizando los bordes en el horizonte y el punto a agregar. Las nuevas facetas son calculadas en la línea 21 y mostradas en la Figura 4.8(g). Como último paso la nueva envoltura convexa es calculada, mediante la unión del conjunto de facetas no visibles (Figura 4.8(c)) y las nuevas facetas calculadas (Figura 4.8(g)); línea 22. El resultado se muestra en la Figura 4.8(h).

Todos los cálculos en el Algoritmo 9 se realiza utilizando únicamente el tipo de datos arreglo, el algoritmo no requiere ningún cambio para utilizarse en dimensiones más altas. En el caso de que un punto sea interior, el conjunto de facetas visibles es el conjunto vacío, lo cual implica que todas las intersecciones calculadas resultan en el conjunto vacío, incluyendo el horizonte y el conjunto de nuevas facetas. En este caso el resultado de una iteración del Algoritmo 9 será el conjunto de facetas no visibles, es decir, la envoltura convexa no sufre cambio alguno.

4.7. Conclusiones del capítulo

Los algoritmos genéticos y la optimización de enjambre de partículas no garantizan la completitud ni la correctitud cuando se aplican en problemas multimodales [Li J.-P.02, Li06]. La mayoría de los algoritmos dependen fuertemente de valores de parámetros que no son conocidos a priori, como el radio. Estos parámetros son determinantes en la operación correcta de los algoritmos. Incluso teniendo un valor óptimo para los parámetros no hay garantía de encontrar todos los óptimos o que todos los óptimos reportados sean realmente

óptimos de la función objetivo. Hemos probado que nuestra aproximación provee ambas correctitud y completitud, aunque el análisis de la complejidad muestre un comportamiento exponencial con respecto a la dimensión del problema, y sea necesario un gran número de puntos, comparado con algoritmos genéticos u optimización de enjambre de partículas.

Uno de los temas de investigación actual se enfoca en la eliminación de la dependencia del parámetro del radio, de los algoritmos evolutivos para problemas multimodales. Una de las principales desventajas del uso de parámetro del radio se presenta al no conocer la distancia entre un óptimo y otro, lo cual tiene como consecuencia que si dos óptimos están a una distancia menor que el radio, los algoritmos no podrán distinguir los dos óptimos y serán considerados como uno solo. Además si el radio es muy pequeño, este tendrá un efecto negativo en el desempeño del algoritmo, al aumentar el número de regiones a calcular y dando óptimos espurios.

En este capítulo se ha presentado un algoritmo de optimización que no tiene dependencia en parámetros como el radio. El algoritmo del explorador, como su nombre lo indica, explora el espacio de búsqueda mediante un muestreo inicial de puntos. El algoritmo del explorador está diseñado para problemas de optimización con funciones continuas, sin embargo, utiliza una representación discreta de éstas; por lo cual no debe ser difícil adaptar el algoritmo del explorador a funciones discretas.

Debe considerarse que el algoritmo del explorador depende de la función de distancia del espacio de búsqueda. No es posible utilizarlo en espacios de búsqueda que no tengan una función de distancia bien definida. Además está diseñado para problemas donde la posición de los óptimos no cambia. La implementación actual no puede ser utilizada en problemas de entorno dinámico. No existe ninguna referencia a algoritmos semejantes al que se presenta, por lo cual éste es considerado como una aportación en el área de optimización.

Además, dentro del área de geometría computacional, se presenta una representación para facetas y bordes que simplifica la implementación del algoritmo incremental de envoltura convexa en altas dimensiones utilizando un mínimo de recursos de memoria. Se presenta también una implementación del algoritmo que funciona en cualquier dimensión.

Capítulo 5

Algoritmos de Inteligencia Artificial en Sistemas Dinámicos

En este capítulo se presenta la implementación de los algoritmos de inteligencia artificial presentados en los dos capítulos anteriores, para su uso en el problema de búsqueda de soluciones en ecuaciones no lineales y en la generación de diagramas de bifurcación.

5.1. Algoritmos genéticos

Para la representación de los individuos en los algoritmos genéticos se optó por utilizar números de punto flotante en lugar de secuencias de caracteres [Randy L. Haupt04, Aggarwal00], dado que se optimizarán funciones de valores reales. Además como es necesario representar cada una de las variables de las ecuaciones se optó por representar el cromosoma como un vector con elementos de punto flotante donde cada índice representa a una variable [Randy L. Haupt04]. Así, si tenemos las variables de estado de un sistema dinámico $v_1, v_2, \ldots, v_n$ el cromosoma que representa al individuo a será

$$cromosoma(a) = (v_1, v_2, \ldots, v_n)$$

Con esta representación para el cromosoma de los individuos cada uno de ellos representa también un punto en el espacio fase del sistema dinámico. No obstante, debido a que el desarrollo de los algoritmos genéticos está enfocado a la representación de los individuos por medio de secuencias de caracteres, es necesario adaptar muchos de los métodos a

la representación de vectores de punto flotante utilizada en esta tesis [Randy L. Haupt04]. Principalmente los métodos para los operadores de cruza y mutación deben realizarse de manera adecuada para evitar que los individuos salgan del espacio de búsqueda. Los métodos están descritos en las siguientes secciones.

5.1.1. Cruza

La cruza de dos individuos en la codificación utilizada se efectúa de la forma siguiente: se selecciona un número aleatorio q entre 0 y 1, y definimos $p = 1 - q$. Para la pareja de individuos a, b se generan hijos h_1 y h_2 utilizando las ecuaciones

$$\boldsymbol{h}_1 = p\boldsymbol{a} + q\boldsymbol{b} \tag{5.1}$$

$$\boldsymbol{h}_2 = q\boldsymbol{a} + p\boldsymbol{b} \tag{5.2}$$

donde $\boldsymbol{a}$ y $\boldsymbol{b}$ son los vectores que corresponden a los padres y $\boldsymbol{h_1}$ y $\boldsymbol{h_2}$ son los vectores que serán los cromosomas de los hijos.

Para verificar que los nuevos individuos $\boldsymbol{h}_1$ y $\boldsymbol{h}_2$ están dentro del espacio de búsqueda, primero se hace la suposición de que uno de los nuevos individuos está fuera del espacio de búsqueda; al menos una de sus coordenadas sobrepasa su valor máximo. Si l_{mi} es el valor máximo para la coordenada i y el hijo $\boldsymbol{h}_1$ esta fuera del espacio de búsqueda sobrepasando el valor máximo de la coordenada i, tenemos que

$$l_{mi} < p\boldsymbol{a}_i + q\boldsymbol{b}_i \tag{5.3}$$

Como $\boldsymbol{a}$ y $\boldsymbol{b}$ sí están dentro del espacio de búsqueda, entonces se cumplen las relaciones

$$\boldsymbol{a}_i < l_{mi} \tag{5.4}$$

$$\boldsymbol{b}_i < l_{mi} \tag{5.5}$$

Si sustituimos $\boldsymbol{a}_i$ y $\boldsymbol{b}_i$ en la Ecuación (5.3), la desigualdad se sigue cumpliendo y tenemos

$$l_{mi} < p\ l_{mi} + q\ l_{mi} \tag{5.6}$$

de donde, factorizando tenemos

$$l_{mi} < (p + q)l_{mi} \tag{5.7}$$

y como definimos al principio de esta sección $p = 1 - q$

$$l_{mi} < (1 - q + q)l_{mi} \tag{5.8}$$

o

$$l_{mi} < l_{mi} \tag{5.9}$$

Dado que la condición de la Ecuación (5.9) es falsa, entonces la suposición de que el nuevo individuo está fuera del espacio de búsqueda es falsa, y se concluye que todos los individuos generados mediante las Ecuaciones (5.1) y (5.2) estarán dentro del espacio de búsqueda.

5.1.2. Mutación

La mutación para la representación de vectores de punto flotante, se realiza de manera muy similar a como se realiza para una representación binaria [Randy L. Haupt04, Goldberg00]; se selecciona aleatoriamente una coordenada del vector que representa al individuo y al contenido se le suma un valor aleatorio. Debe tenerse en cuenta que al sumar un valor aleatorio a una coordenada, ésta puede quedar fuera del rango del espacio de búsqueda, además si el valor de la coordenada se sale de rango, y solo se trunca el valor de la coordenada al límite del rango, pueden generarse anomalías estadísticas (todas las mutaciones terminarían en cambiar el valor de una coordenada a los límites de su rango). Para evitar cualquier anomalía se optó por desplazar la coordenada al origen. La cantidad que se suma a una coordenada i seleccionada, es calculada aleatoriamente dentro del rango de $[0, M_i]$, donde M_i es la magnitud del rango de búsqueda para la coordenada i. Después de sumar al valor de la coordenada i una cantidad aleatoria, se aplica la función módulo a la coordenada i con divisor M_i, para evitar que el nuevo valor salga del rango búsqueda. El algoritmo para el operador de mutación es presentado en el Algoritmo 10.

Algoritmo 10 mutacion(vector, rangos)

```
1: longitud ← length(vector)
2: i ← random() * longitud
3: magnitud ← abs(rangos[i][1] - rangos[i][0])
4: valor_temporal ← vector[i] - rangos[i][0]
5: valor_aleatorio ← random() * magnitud
6: valor_temporal ← (valor_temporal + valor_aleatorio) mod magnitud
7: vector[i] ← valor_temporal + rangos[i][0]
```

5.2. Optimización de enjambre de partículas

A diferencia de los algoritmos genéticos, en el algoritmo de optimización de enjambre de partículas, cada partícula es representada por una posición en el espacio de búsqueda (representación en variables continuas). Al igual que en el caso de algoritmos genéticos, la posición de una partícula estará representada por un vector con elementos de puntos flotante, si tenemos las variables de estado de un sistema dinámico $v_1, v_2, \ldots, v_n$ la posición que representa al punto p será

$$posición(p) = (v_1, v_2, \ldots, v_n)$$

Al ser enfocado el algoritmo de optimización de enjambre de partículas a vectores de numeros de punto flotante, no es necesario modificar los métodos para calcular la velocidad y posición los cuales se mostraron en la Sección 3.2

5.3. Criterio de terminación para algoritmos evolutivos

En la presente tesis los algoritmos evolutivos se iteran un número determinado de generaciones y se examinan los resultados obtenidos; se da un criterio de error basado en la magnitud del vector que representa al individuo. En los problemas donde se necesita encontrar una solución de un sistema de ecuaciones, el objetivo es obtener vectores resultantes tan cercanos a cero como sea posible, la medida más natural para saber que tan cerca estamos de cero es calcular la norma del vector. El error se define entonces como la magni-

tud del vector resultante después de aplicar la función objetivo al vector que representa un individuo.

5.4. Búsqueda de soluciones de un sistema de ecuaciones no lineales como un problema de optimización

En algoritmos genéticos y optimización de enjambre de partículas es necesario que la función objetivo regrese un sólo valor, no un vector, y que este valor sea no negativo [Goldberg00, Kennedy01]. Para cumplir con este requisito se utiliza una función de mapeo [Goldberg00]. La función de mapeo puede ser una composición de varias funciones, pero cada función en la composición debe preservar los óptimos de la función objetivo.

En el caso de la búsqueda de soluciones para un sistema de ecuaciones no lineales, el problema general que tenemos que resolver es el representado en la Ecuación 5.10

$$\mathbf{f}(x, y, \alpha) = 0 \tag{5.10}$$

donde, si el sistema de ecuaciones no lineales $\mathbf{f}$ es el sistema de eacuaciones algebraico asociado a un sistema dinámico $x \in \mathbb{R}^n$ representa un punto en el espacio fase del sistema dinámico, $y \in \mathbb{R}^m$ representa las variables del sistema dinámico no asociadas al espacio fase (conjunto de restricciones) y $\alpha \in \mathbb{R}^k$ representa el conjunto de parámetros asociados al sistema dinámico. En el Capítulo 2 se da una descripción del problema de la búsqueda de condiciones iniciales sin expresar de manera explícita las restricciones que puede tener un sistema dinámico. De manera general, los sistemas dinámicos se estudian considerando las restricciones de manera implícita [Kuznetsov98, Strogatz00], i.e. sólo se considera el vector $x \in \mathbb{R}^n$ como un punto del espacio fase, y el vector de parámetros $\alpha \in \mathbb{R}^k$.

Para la búsqueda de soluciones de sistemas de ecuaciones no lineales de la forma de la Ecuación (5.10) utilizando algoritmos evolutivos, la función $\mathbf{f}$ es tomada como la función objetivo; sólo se considera al punto $x \in \mathbb{R}^n$ para formar el cromosoma que representa a un individuo en los algoritmos evolutivos como se indica en las Secciones 5.1 y 5.2. El conjunto de restricciones $y \in \mathbb{R}^m$ son consideradas dentro de la función objetivo y los parámetros asociados al sistema dinámico $\alpha \in \mathbb{R}^k$ permanecen constantes dentro del ciclo de los algoritmos evolutivos.

Dado que no se puede utilizar directamente el sistema de ecuaciones no lineales como función de aptitud para los algoritmos genéticos ni para optimización de enjambre partículas, fue necesario utilizar una función de mapeo. Debido a que el problema es encontrar soluciones de un sistema de ecuaciones no lineales, la función de mapeo natural es la norma del vector resultante al evaluar el punto que representa a un individuo en la función objetivo, aunque no es la única función de mapeo que se puede utilizar. En la siguiente sección se describen las funciones de mapeo que se utilizaron en la presente tesis.

5.4.1. Función de error

En los problemas donde se necesita encontrar una solución de un sistema de ecuaciones, el objetivo es obtener vectores resultantes tan cercanos a cero como sea posible, en la presente tesis se toma como valor mínimo de error 10^{-6}. Para evaluar el desempeño de los algoritmos evolutivos se define la siguiente función de error

Definición 3 *Dado el sistema de ecuaciones no lineales representado por la Ecuación (5.10), definimos el error en el punto $x \in \mathbb{R}^n$ del espacio de búsqueda como*

$$error(x) = |\mathbf{f}(x, y, \alpha)| \tag{5.11}$$

donde $|\mathbf{f}(x, y, \alpha)|$ es la norma estándar de $\mathbf{f}(x, y, \alpha)$ en $\mathbb{R}^n$ [Rudin76, Apostol74].

Definición 4 *La **norma** $|\mathbf{f}(x, y, \alpha)|$, para $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$ y $\alpha \in \mathbb{R}^k$ está definida como*

$$|\mathbf{f}(x, y, \alpha)| = \left(\sum_{i=1}^{n} f_i(x, y, \alpha)^2 \right)^{1/2} \tag{5.12}$$

5.4.2. Función de mapeo

La primera función de mapeo es la norma del vector resultante, i.e., tenemos la función de mapeo

$$h(x) = |\mathbf{f}(x, y, \alpha)| \tag{5.13}$$

donde $\mathbf{f}$ es el sistema de ecuaciones no lineales del cual se buscan las soluciones (v.g. el sistema de Ecuaciones (6.1) a (6.4)). la función de mapeo $h(x)$ sólo regresa un valor y es no negativo.

La segunda función de mapeo es la descrita en la Ecuación (5.14) presentada en el artículo de Varum Aggarwal [Aggarwal00], el cual propone una transformación para convertir el problema de búsqueda de soluciones (puntos x para los cuales $\mathbf{f}(x) = 0$), en una búsqueda de máximos. Ésto se logra mediante la aplicación de la siguiente ecuación:

$$g(x) = \frac{1}{1 + |\mathbf{f}(x, y, \alpha)|} \tag{5.14}$$

En el artículo de Aggarwal este mapeo se usó en conjunto con el algoritmo genético de tipo "niching" desarrollado por Alan Petrowski [Petrowski97], en el que se divide a la población en subpoblaciones, y dónde sólo el individuo con la aptitud máxima de la subpoblación es candidato a reproducción.

De la ecuación (5.14) es fácil ver que para toda x tal que $|\mathbf{f}(x, y, \alpha)| = 0$ el valor de $g(x)$ es de 1. Para cualquier otro valor de $|\mathbf{f}(x, y, \alpha)|$ diferente de cero el valor de $g(x)$ será menor que 1, pero sin llegar a ser 0, i.e., $g(x) \to 0$ cuando $|\mathbf{f}(x, y, \alpha)| \to \infty$. Ésto es, el problema de buscar soluciones para $\mathbf{f}(x, y, \alpha)$ se convierte en la búsqueda de valores máximos de $g(x)$ y cumple los requisitos de una función de aptitud.

Como ejemplo del uso de la función de mapeo g (Ecuación (5.14)), en el sistema dinámico descrito en la Sección 6.1 la función $\mathbf{f}$ consiste en el sistema de ecuaciones formado por las Ecuaciones (6.1) a (6.4). Dado que las funciones de mapeo sólo modifican el valor de aptitud de los individuos dentro del algoritmo genético, éstas no modifican el valor de la función de error para un individuo (definida en la Sección 5.4.1).

5.5. Trazo de diagramas de bifurcación como un problema de optimización en entorno dinámico

El problema de realizar el trazo de diagramas de bifurcación completos para un sistema dinámico puede plantearse de igual forma que el problema de la búsqueda de soluciones de un sistema de ecuaciones no lineales, sin embargo, en este caso las soluciones del sistema de ecuaciones no lineales (los óptimos que son buscados), pueden cambiar su posición, aparecer, desaparecer, o presentar otro tipo de comportamiento con respecto a la

variación de alguno de los parámetros del sistema [Kuznetsov98, Strogatz00].

El comportamiento que pueden exhibir los puntos fijos de un sistema dinámico (las soluciones de un sistema de ecuaciones no lineales) con respecto al cambio de un parámetro, es semejante al estudiado en optimización en el llamado problema de entorno dinámico. En un problema de entorno dinámico los óptimos de una función presentan cambios en su posición, en su calidad de óptimo local o global, o incluso desaparecen, con respecto al cambio en un parámetro [Li06]. El caso más simple es la traslación de un óptimo con respecto a un parámetro de tiempo.

Se eligió utilizar solamente el algoritmo de optimización de enjambre de partículas con especies desarrollado por Parrot [Parrott06], debido a que los algoritmos genéticos para entornos dinámicos no presentan buenos resultados [Li J.-P.02] y el algoritmo del explorador no fue diseñado en específico para este tipo de problemas de optimización.

Para plantear el problema de trazo de diagramas de bifurcación completos como un problema de optimización en un entorno dinámico comenzamos con una breve descripción del problema de trazo diagramas de bifurcación. Dado el conjunto de ecuaciones diferenciales que modela un sistema dinámico con un solo parámetro

$$\dot{x} = \mathbf{f}(x, y, \alpha),\ x \in \mathbb{R}^n,\ y \in \mathbb{R}^m,\ \alpha \in \mathbb{R}^k \tag{5.15}$$

La forma estándar de generar un diagrama de bifurcación [Kuznetsov98, Govaerts00, Eusebius97, Ermentrout06, Doedel97] consiste en encontrar un punto fijo estable, y después, usando un método de continuación se determina el siguiente punto fijo, basado en la condición

$$\mathbf{f}(x, y, \alpha) = 0 \tag{5.16}$$

la cual define una curva k-dimensional suave en $\mathbb{R}^{n+k}$. Nuestra aproximación a este problema es determinar todos los puntos $x \in X$ para una región dada $X \subset \mathbb{R}^n$ que satisfacen la Ecuación (5.16) con el conjunto de restricciones $y \in \mathbb{R}^m$, para un vector dado de valores para los parámetros $\alpha \in \mathbb{R}^k$.

Para encontrar todos los puntos x, usando el algoritmo de optimización de enjambre de partículas, primero transformamos el problema de encontrar todas las soluciones para la Ecuación (5.16) en un problema de encontrar todos los máximos usando la función de mapeo dada por la Ecuación (5.14), como se menciona en la sección anterior.

Comenzando con un vector de valores $\alpha = \alpha_0$, se aplica el algoritmo de optimización de enjambre de partículas un número dado de iteraciones para encontrar todos los máximos de la función g con el valor fijo α_0. Una vez que son encontrados todos los puntos fijos x para una región dada y un vector de valores para los parámetros fijos α_0, éstos son almacenados, entonces el vector de parámetros α es cambiado a $\alpha_1 = \alpha_0 + \Delta\alpha$ y un nuevo conjunto de puntos fijos x es determinado; la nueva búsqueda está basada en la información acumulada en el enjambre anterior. Basados en lo anterior, el diagrama de bifurcación es generado cambiando los valores de los parámetros α y almacenando todas las soluciones encontradas para cada vector de valores α_t de los parámetros.

La estabilidad de los puntos obtenidos es determinada por el análisis de los valores propios de la matriz del Jacobiano para el sistema algebraico de ecuaciones evaluado en el punto. Los elementos de la matriz del Jacobiano son determinados utilizando diferenciación numérica mediante el algoritmo de diferenciación de Richardson [Kincaid91]; los valores propios son calculados primero transformando la matriz del Jacobiano en su forma Hessemberg superior y después usando el algoritmo QR para obtener los valores propios de la matriz del Jacobiano. El algoritmo de QR está descrito en la Sección 2.2, una descripción más completa de los algoritmos numéricos auxiliares puede encontrarse en [Kincaid91].

5.6. Observaciones

En el caso del algoritmo del explorador presentado en esta tesis, se utiliza la misma representación que en optimización de enjambre de partículas. Este algoritmo fue desarrollado de manera particular para optimización de funciones de valores reales, además de estar directamente relacionado con un algoritmo de geometría computacional conocido como el algoritmo incremental de envoltura convexa [Franco P. Preparata85].

En el caso de la búsqueda de soluciones de sistemas de ecuaciones no lineales, las restricciones son introducidas en la función objetivo; una población generada para algoritmos genéticos u optimización de enjambre de partículas sólo tiene como restricciones los límites del espacio de búsqueda [Goldberg00].

5.7. Algoritmos auxiliares

Se utilizaron algoritmos auxiliares con el propósito de mejorar la precisión de los puntos obtenidos por medio de los algoritmos evolutivos. También se usarán algoritmos auxiliares para calificar la estabilidad de los puntos obtenidos. En las siguientes secciones son descritas las implementaciones que fueron utilizadas para cada algoritmo.

5.7.1. Algoritmo de Levenberg-Marquardt

El algoritmo de Levenberg-Marquardt tiene su origen en la solución de problemas de mínimos cuadrados para ajuste de curvas [Levenberg44, Marquardt63, Nocedal99, William H. Press92]: dado un conjunto empírico de pares de datos (t_j, y_j), optimizar los parámetros p de la curva modelo $f(t|p)$ de tal manera que la suma de los cuadrados de las desviaciones

$$S(p) = \sum_{i=1}^{m} [y_i - f(t_i|p)]^2 \tag{5.17}$$

sea mínima.

De manera semejante a otros algoritmos numéricos para minimización, el algoritmo de Levenberg-Marquardt es un proceso iterativo. Para comenzar la minimización, el usuario debe proveer un valor inicial para el vector de parámetros p. En muchos casos un punto inicial estándar $p^T = (1, 1, \ldots, 1)$ funciona bien; en otros casos el algoritmo converge solo si el punto inicial es de alguna forma cercano a la solución final.

En cada paso de la iteración, el vector de parámetros p es reemplazado por un nuevo estimado $p + q$. Para determinar q, las funciones $f(p + q)$ son aproximadas por sus versiones linealizadas

$$f(p + q) \approx f(p) + Jq \tag{5.18}$$

donde J es el Jacobiano de f evaluado en q.

En un mínimo de la suma de cuadrados S, el gradiente de S con respecto de q es cero. Diferenciando el cuadrado del lado derecho de la Ecuación (5.17) e igualándola a cero tenemos:

$$(J^T J)q = J^T[y - f(p)] \tag{5.19}$$

de donde q puede ser obtenida calculando la inversa de $J^T J$. La principal diferencia del algoritmo de Levenberg-Marquardt es el reemplazo de la Ecuación (5.19) por una versión con "amortiguamiento"

$$(J^T J + \gamma I)q = J^T[y - f(p)] \tag{5.20}$$

Donde I es la matriz identidad. De la Ecuación (5.20) podemos obtener q al multiplicar por la matriz inversa de $(J^T J + \gamma I)$ en ambos lados de la ecuación, y obtenemos

$$q = (J^T J + \gamma I)^{-1} J^T[y - f(p)] \tag{5.21}$$

El factor de amortiguamiento γ es ajustado en cada iteración. Iniciando con un valor de 10^{-3}, si la reducción de S es rápida, se utiliza un valor más pequeño de γ, típicamente un orden de magnitud menor. Ésto acerca al algoritmo de Levenberg-Marquardt al comportamiento del algoritmo de Gauss-Newton. De otra forma, si la reducción de S no es suficientemente grande γ puede ser incrementado, igual que con la reducción éste se incrementa un orden de magnitud; ésto hace que el algoritmo de Levenberg-Marquardt se comporte como un algoritmo de descenso de gradiente.

Si la magnitud del paso q o la reducción de la suma de cuadrados calculados a partir del último vector de parámetros $p + q$ es más pequeño que los valores de límites predefinidos, la iteración se aborta y el último vector de parámetros q es considerado como la solución.

No hay un criterio establecido para la selección del valor inicial de γ, aunque se sugiere que se inicie con un valor de $\gamma < 1$ y éste sea incrementado o decrementado en factores de 10.

5.7.2. Determinación de la estabilidad de un punto fijo

Una forma de determinar la estabilidad de un punto fijo, es analizando los valores propios del Jacobiano de las ecuaciones algebraicas, evaluado en el punto. Si alguno de los

valores propios presenta parte real positiva, el punto se considera inestable; de otra forma, el punto es considerado como estable.

Para determinar la estabilidad de un punto fijo es necesario calcular los valores propios de la matriz que representa el Jacobiano, donde el tamaño de la matriz es igual al número de dimensiones del espacio de búsqueda. De los algoritmos existentes se optó por el algoritmo iterativo de QR para la diagonalización de una matriz; debido a que es el estándar para la mayoría de las aplicaciones existentes por su grado de precisión, y mejor tiempo de convergencia [Kincaid91]. Además se utilizaron en conjunto los métodos de transformación la matriz en la forma Householder para reducir la matriz a una forma triangular superior, y el método de desplazamiento en la iteración, QR, para mejorar la velocidad de convergencia y reducción de la matriz. Este método reduce la dimensión de la matriz dentro del ciclo de la iteración, lo cual a su vez reduce el número de iteraciones a realizar.

El algoritmo de QR es un algoritmo iterativo diseñado para revelar los valores propios de una matriz A. Haciendo uso del teorema de Schur, de acuerdo al cual toda matriz cuadrada es similar unitaria a un matriz triangular. Ésto es, la matriz A tiene una factorización del tipo

$$UAU^* = T \tag{5.22}$$

donde U es una matriz unitaria, i.e. cumple con la condición $UU^* = I$, y T es triangular. Como los valores propios de A y T son iguales y dado que los valores propios de una matriz triangular son simplemente los elementos de la diagonal, podemos encontrar los valores propios de A en la diagonal de T. Aunque sabemos que existe la factorización de la matriz A, no es simple de calcular y es comparable en dificultad a encontrar todas las raíces (complejas) de un polinomio de grado n para una matriz de dimensiones $n \times n$.

Como lo indica el nombre del algoritmo, éste usa la factorización QR para una matriz y por lo tanto solo se puede usar con matrices cuadradas. La factorización QR consiste en escribir la matriz A en la forma

$$A = QR \tag{5.23}$$

donde Q es unitaria y R es triangular superior. El algoritmo de QR en su forma más básica es como se presenta en el Algoritmo 11.

Algoritmo 11 iteracion_qr(A)

1: $A_1 \leftarrow A$
2: **for** k = 1, 2,. . . , M **do**
3: $[Q_k, R_k] \leftarrow$ **factorización_householder**(A_k)
4: $A_{k+1} \leftarrow R_k Q_k$
5: **end for**
6: **return** A_M

Si las circunstancias son favorables, cuando $k \rightarrow \infty$, la diagonal de A_k converge, a un vector cuyos componentes son los valores propios de A. Primero se debe observar que todas las matrices A_k producidas en el Algoritmo 11 son unitarias similares a A, esto se puede ver de la ecuación

$$A_k = Q_k R_k = (Q_k R_k)(Q_k Q_k^*) = Q_k A_{k+1} Q_k^* \tag{5.24}$$

en concordancia con el teorema de Schur. Así A_k converge a una matriz diagonal con los valores propios de A. Segundo, debemos notar que si A es una matriz real entonces todas las matrices A_k serán reales; si una matriz contiene valores propios con parte imaginaria diferente de cero, lo único que se puede esperar bajo las mejores circunstancias es que A_k converja a una matriz "triangular" con submatrices de 2×2 en su diagonal.

En su forma básica, el algoritmo QR puede tener problemas de velocidad de convergencia para algunas matrices [Kincaid91]. Esto se resuelve utilizando un "desplazamiento" del origen, este desplazamiento es definido como el reemplazo de la matriz A por $A - zI$ [Kincaid91]. El algoritmo QR con desplazamiento es realizado de la siguiente manera.

Si el escalar z_k es tomado como el elemento inferior derecho de la diagonal de A_k entonces la iteración producirá de forma rápida un vector de la forma $(0, 0, \ldots, 0, \alpha)^T$ en la última columna. El número α es entonces un valor propio de la matriz A. La mejor forma de proceder después de que se llega a este vector es reducir la matriz descartando el último renglón y la última columna. Repetir el procedimiento de desplazamiento con la matriz resultante. El proceso de reducción es descrito a continuación.

Si una matriz es escrita en la siguiente forma

$$A = \begin{bmatrix} B & C \\ 0 & E \end{bmatrix} \tag{5.25}$$

donde B y E son matrices cuadradas, entonces el conjunto de valores propios de A es igual a la unión del conjunto de valores propios de B y el conjunto de valores propios de E. De la Ecuación (5.25) podemos escribir $Ax = \lambda x$ como

$$\begin{bmatrix} B & C \\ 0 & E \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \lambda \begin{bmatrix} u \\ v \end{bmatrix} \tag{5.26}$$

O de manera equivalente

$$\begin{aligned} Bu + Cv &= \lambda u \\ Ev &= \lambda v \end{aligned} \tag{5.27}$$

Si α es un valor propio de A entonces la Ecuación (5.27) tiene una solución $(u, v)^t$ no trivial. Esto es, u o v pueden ser cero pero no los dos al mismo tiempo. Si $v \neq 0$ entonces λ es un valor propio de E. Si $v = 0$ entonces $u \neq 0$ y entonces λ es un valor propio de B. De manera inversa, si λ es un valor propio de B y u es su vector propio correspondiente, entonces $(u, 0)^T$ resuelve la Ecuación (5.27) y λ es un valor propio de A. Si λ es un valor propio de E pero no de B, entonces podemos encontrar un vector v que satisface $Ev = \lambda v$, después podemos resolver $(B - \lambda I)u = -Cv$. Esta ecuación tiene solución debido a que λ no es un valor propio de B y entonces $(u, v)^T$ es una solución de la Ecuación (5.27) y λ es un valor propio de A.

En conjunto, estos algoritmos pueden determinar los valores propios para una matriz con alta precisión y buena velocidad de convergencia. La descripción del algoritmo de Levenberg-Marquardt puede ser encontrada en [Levenberg44] y [Marquardt63]. Una descripción completa de los algoritmos para encontrar los valores propios asociados con una matriz dada pueden ser encontrados en [Kincaid91].

5.8. Conclusiones del capítulo

En este capítulo hemos mostrado la implementación de los algoritmos evolutivos para el problema de la búsqueda de puntos fijos para un sistema dinámico y para el trazo

de diagramas de bifurcación. Al mismo tiempo se muestra como son utilizado los algoritmos evolutivos en problemas como la búsqueda de soluciones. También se ha definido la función de error, así como los algoritmos auxiliares que serán utilizados en el siguiente capítulo donde se examinan los casos de estudio.

Capítulo 6

Casos de estudio

En el presente capítulo se presentan los resultados obtenidos mediante la aplicación de los algoritmos de inteligencia artificial, así como el algoritmo desarrollado en el Capítulo 4, a los problemas presentados en el Capítulo 2. Los resultados son presentados siguiendo los pasos del análisis de sistemas dinámicos: primero se muestran los resultados obtenidos para la búsqueda de puntos fijos de sistemas dinámicos, se muestra una comparativa de varios algoritmos, además se presentan los resultados obtenidos con el algoritmo desarrollado. Después son mostrados los resultados de la generación de diagramas de bifurcación con el algoritmo de optimización de enjambre de partículas. Se incluyen resultados en sistemas con uno y dos parámetros.

6.1. Red eléctrica de tres nodos

La red eléctrica mostrada en la Figura 6.1 ha sido usada en [K.86, Ian Dobson88] y en otros artículos de investigación [Abed84, Kwanty86, Ajjarapu92, Abed92, Chiang94, Cañizares95, D.J.95] para ilustrar el fenómeno de colapso de voltaje que se presenta con la variación de la carga. El circuito equivalente de tres nodos debe ser visto como un circuito equivalente a un área local de interés, conectado a una red mayor. La red es modelada como un bus infinito representado por una fuente de voltaje que provee un voltaje constante en magnitud y fase $E_0\angle\theta_0$ sin importar el flujo de potencia. $E_m\angle\delta_m$ es la magnitud del voltaje interno del generador, que para este ejemplo corresponde al voltaje en las terminales del mismo. La admitancia compleja de las líneas de transmisión conectadas al generador y al bus infinito, una carga y un capacitor son mostradas en la Figura 6.1. El voltaje medido en

la terminal es $V\angle\delta$.

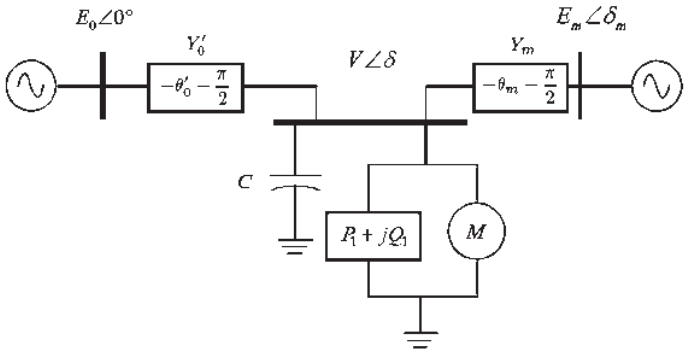


Figura 6.1: Diagrama del sistema eléctrico de potencia.

La dinámica del sistema está gobernada por las siguientes cuatro ecuaciones diferenciales ordinarias [K.86, Ian Dobson88]

$$\dot{\delta}_m = \omega \tag{6.1}$$

$$\dot{\omega} = \frac{1}{M}(P_m - D_m\omega + VE_mY_m\sin(\delta - \delta_m - \theta_m) + E_m^2Y_m\sin\theta_m) \tag{6.2}$$

$$\dot{\delta} = \frac{1}{K_{qw}}(-K_{qv2}V^2 - K_{qv}V + Q - Q_0 - Q_1) \tag{6.3}$$

$$\dot{V} = \frac{1}{TK_{qw}K_{pv}}[-K_{qw}(P_0 + P_1 - P) + (K_{pw}K_{qv} - K_{qw}K_{pv})V + K_{pw}(Q_0 + Q_1 + Q) + K_{pw}K_{qv2}V^2] \tag{6.4}$$

En estas ecuaciones las variables de estado están definidas como sigue: δ_m es el ángulo de fase del voltaje del generador, ω es la velocidad del rotor, δ es el ángulo la fase de la carga de voltaje, y V es la magnitud del voltaje de carga. Las funciones P y Q que aparecen en estas ecuaciones representan, respectivamente, la potencia activa y reactiva proporcionadas por la carga de la red. Éstas son dadas por las Ecuaciones (6.5) y (6.6) [K.86, Ian Dobson88]

$$
\begin{aligned}
P &= -VE_0'Y_0'\sin(\delta+\theta_0') - VE_mY_m\sin(\delta-\delta_m+\theta_m) \\
&\quad +V^2(Y_0'\sin\theta_0' + Y_m\sin\theta_m) \qquad (6.5)\\
Q &= VE_0'Y_0'\cos(\delta+\theta_0') + VE_mY_m\cos(\delta-\delta_m+\theta_m) \\
&\quad -V^2(Y_0'\cos\theta_0' + Y_m\cos\theta_m) \qquad (6.6)
\end{aligned}
$$

En las ecuaciones anteriores, en lugar de incluir el capacitor C en el circuito, un circuito equivalente de Thevenin con un capacitor fue derivado con los valores dados por las Ecuaciones (6.7), (6.8) y (6.9)

$$
\begin{aligned}
E_0' &= \frac{E_0}{\sqrt{1+C^2Y_0^{-2}-2CY_0^{-1}\cos\theta_0}} \qquad (6.7)\\
Y_0' &= Y_0\sqrt{1+C^2Y_0^{-2}-2CY_0^{-1}\cos\theta_0} \qquad (6.8)\\
\theta_0' &= -\frac{\pi}{2}+\tan^{-1}\frac{C-Y_0\cos\theta_0}{-Y_0\sin\theta_0} \qquad (6.9)
\end{aligned}
$$

Las otras cantidades que aparecen en las ecuaciones son parámetros constantes, relacionados a la carga o la red y el generador. Todos estos parámetros tienen valores fijos durante el análisis, excepto por Q_1, la demanda de potencia reactiva de la carga. Los parámetros de la carga, la red y el generador son dados en la Tabla 6.1.

Tabla 6.1: Valores constantes para los parámetros en el sistema de ecuaciones diferenciales.

parámetro	valor	parámetro	valor	parámetro	valor
K_{pw}	0.4 pu	K_{pv}	0.3 pu	K_{qw}	-0.03 pu
K_{qv2}	2.1 pu	T	8.5 pu	K_{qv}	-2.8 pu
P_0	0.6 pu	P_1	0.0 pu	Q_0	1.3 pu
E_0'	2.5 rad	P_m	1.0 pu	E_m	1.0 pu
M	0.3 seg^2/rad	Y_m	5.0 pu	Y_0'	8.0 pu
θ_0'	-0.2094 rad	Q_1	10.0 pu	θ_m	-0.08726 rad
D_m	0.05 seg/rad				

6.2. Búsqueda de puntos iniciales para el análisis de sistemas dinámicos

Como se ha mencionado en el Capítulo 2 la primera parte del análisis de sistemas dinámicos mediante el trazo de diagramas de bifurcación es encontrar un punto estable del sistema dinámico. En esta sección se muestra el uso de tres diferentes algoritmos en la búsqueda de un punto fijo de un sistema dinámico. Los tres algoritmos utilizados fueron: algoritmo genético estándar, el algoritmo genético con especies, y el algoritmo de optimización de enjambre de partículas con especies.

En este parte del análisis se utilizó un sistema de potencia descrito en la Sección 6.1, el diagrama que representa al sistema de potencia se muestra en la Figura 6.1. Solo cuatro variables son consideradas: δ_m, ω, δ, y V. Todos los símbolos restantes son considerados constantes, asignándoles los valores que se muestran en la Tabla 6.1.

Para la inicialización de las poblaciones y enjambres de cada algoritmo fueron asignados rangos específicos para cada una de las variables; los valores fueron iguales para todas las pruebas, los valores para los rangos son mostrados en la Tabla 6.2.

Tabla 6.2: Valores para los rangos de búsqueda de las variables.

variable	rango
δ_m	$[0, 1]$ rad
ω	$[-1, 1]$ rad/seg
δ	$[0, 1]$ rad
V	$[0, 2]$ pu

Además fueron consideradas dos funciones de mapeo, las descritas en la Sección 5.4.2, los resultados de cada una son comparados.

Una solución del sistema (encontrada por cualquiera de los tres algoritmos bajo análisis) es $(\delta_m, \omega, \delta, V) = (0.047, 0.0, 0.310, 1.359)$. Aunque todos los algoritmos convergen al mismo resultado, no todos ellos lo hacen de la misma forma para todos los problemas. No muestran tiempos semejantes de convergencia, ni los valores para el error son los mismos. En las siguientes secciones se analiza el desempeño de los diferentes algoritmos en términos de precisión con diferentes valores para el tamaño de la población.

6.2.1. Algoritmo genético

Los parámetros para el algoritmo genético fueron los siguientes: cada ejecución del programa se realizó por 300 generaciones, y fueron realizadas 1000 ejecuciones para calcular los valores de la desviación estándar y de la media aritmética de la función de error. Se usaron poblaciones de tamaños 100, 200, 300, 400 y 500 para comparar los resultados. Los resultados para la desviación estándar con el algoritmo genético simple usando el mapeo con la función h (Ecuación (5.13)) y el mapeo con la función g (Ecuación (5.14)) son mostrados en la Tabla 6.3 y los resultados para la media aritmética del error son mostrados en la Tabla 6.4.

Tabla 6.3: Media aritmética del error para el mejor individuo obtenido para varios tamaños de población, utilizando el algoritmo genético estándar.

Número de individuos	100	200	300	400	500
Media aritmética del error utilizando $h(x)$	5.439	4.418	3.747	3.396	3.179
Media aritmética del error utilizando $g(x)$	2.730	2.004	1.623	1.459	1.319

Tabla 6.4: Desviación estándar para el valor del error para varios tamaños de población para el algoritmo genético estándar.

Número de individuos	100	200	300	400	500
Desviación estándar del error utilizando $h(x)$	1.917	1.623	1.388	1.307	1.192
Desviación estándar del error utilizando $g(x)$	1.362	1.016	0.858	0.809	0.494

Puede observarse de los resultados expresados en las tablas que el tamaño de la población afecta de forma considerable el resultado del error obtenido. Además la función de mapeo que se utiliza también afecta los resultados. Ésto es fácil de observar en los resultados para una tamaño de población de 500, tanto la media aritmética del error, como la desviación estándar son menos de la mitad de los obtenidos con el mapeo de la norma.

6.2.2. Algoritmo genético multimodal

Los parámetros para el algoritmo genético multimodal son los siguientes: el valor para el parámetro del radio es fijado a 1.0, éste valor fue seleccionado mediante experimentación con varios valores para el radio; en cada ejecución del programa se corrieron 300 generaciones; se usaron 1000 ejecuciones para calcular la desviación estándar y la media aritmética del error. Se usaron poblaciones de tamaños 100, 200, 300, 400 y 500 para

comparar los resultados. Los resultados para el cálculo de la desviación estándar usando el mapeo con la función h y la función g son mostrados en la Tabla 6.5 y los resultados para la media aritmética del error son mostrados en la Tabla 6.6.

Tabla 6.5: Media aritmética del error para el mejor individuo obtenido para varios tamaños de población, utilizando el algoritmo genético de conservación de especies.

Número de individuos	100	200	300	400	500
Media aritmética del error utilizando $h(x)$	5.186	3.758	3.155	2.811	2.645
Media aritmética del error utilizando $g(x)$	3.408	2.450	1.932	1.671	1.465

Tabla 6.6: Desviación estándar para el valor del error para varios tamaños de población para el algoritmo genético de conservación de especies.

Número de individuos	100	200	300	400	500
Desviación estándar del error utilizando $h(x)$	1.712	1.387	1.211	1.095	0.992
Desviación estándar del error utilizando $g(x)$	1.696	1.293	1.066	0.922	0.853

Al utilizar el algoritmo genético de conservación de especies no se presentan mejoras considerables con respecto al algoritmo genético estándar, incluso el algoritmo genético de conservación de especies presenta un valor de la desviación estándar más alto en el caso del mapeo con la función $g(x)$. El algoritmo genético de conservación de especies fue diseñado para encontrar más de un óptimo de una función dada, y éste presenta un parámetro adicional que debe ser inicializado: el radio. El valor que se asigne al radio tiene un efecto directo en el desempeño del mismo [Li J.-P.02].

6.2.3. Optimización de enjambre de partículas

Los parámetros para el algoritmo de optimización de enjambre de partículas fueron los siguientes: el valor para el parámetro del radio es fijado en 1.0; en cada ejecución se corrieron 300 iteraciones; se utilizarón 1000 corridas para el cálculo del valor de la desviación estándar y la media aritmética del error. Se utilizarón enjambres con tamaños de 100, 200, 300, 400 y 500 partículas para comparar los resultados. Los resultados para la desviación estándar utilizando el algoritmo de optimización de enjambre de partículas para la función h y la función g son mostrados en la Tabla 6.7 y los resultados para la media aritmética del error son mostrados en la Tabla 6.8.

En el uso del algoritmo de optimización de enjambre de partículas se obtuvieron mejores resultados que con el algoritmo genético estándar y de conservación de especies,

Tabla 6.7: Media aritmética del error para el mejor individuo obtenido para varios tamaños de enjambre, utilizando el algoritmo de optimización de enjambre de partículas.

Número de individuos	100	200	300	400	500
Media aritmética del error utilizando $h(x)$	0.082	0.029	0.013	0.006	0.007
Media aritmética del error utilizando $g(x)$	0.092	0.032	0.012	0.008	0.005

Tabla 6.8: Desviación estándar para el valor del error para varios tamaños de enjambre para el algoritmo de optimización de enjambre de partículas.

Número de individuos	100	200	300	400	500
Desviación estándar del error utilizando $h(x)$	0.211	0.094	0.058	0.038	0.039
Desviación estándar del error utilizando $g(x)$	0.246	0.113	0.051	0.040	0.028

incluso para tamaños de población pequeños, llegando a un error de un orden de 10^{-3} y una desviación estándar de dos centésimas. También se observa que al utilizar la función de mapeo g se obtienen mejores resultados, aunque la mejora no es tan considerable como en el caso de los algoritmos genéticos. Este algoritmo además está diseñado para encontrar más de un óptimo de la función dada, pero requiere de dos parámetros adicionales, el radio y el número de partículas por especie que necesitan calcularse por medio de experimentación o por medio de métodos auxiliares [Li06]. La asignación de un valor adecuado de estos parámetros no es trivial y muchas veces requiere de experimentación o métodos externos para su cálculo, además de que éstos afectan el desempeño del algoritmo [Li06].

Como se ha mostrado con los resultados obtenidos, el algoritmo de optimización de enjambre de partículas brinda mejores resultados, pero debe tenerse en cuenta que necesita de dos parámetros extra que deben ser inicializados. Cada uno de los algoritmos presenta diferentes características.

El algoritmo genético estándar solo requiere de inicializar el número de individuos en la población inicial, no es necesario tener información adicional de la función a optimizar, puede obtenerse una mejora considerable utilizando una función de mapeo como la presentada en la Ecuación (5.14), y puede usarse en conjunto con un algoritmo numérico de gradiente, utilizando el óptimo encontrado como punto inicial para el algoritmo numérico. Aunque el algoritmo genético estándar solo permite obtener un solo óptimo, es posible realizar una nueva búsqueda adicional descartando la región donde ya ha sido encontrado un óptimo.

El algoritmo genético de conservación de especies es un algoritmo diseñado para

encontrar más de un óptimo en una función determinada. Según los resultados, no presenta mejoras considerables en la precisión. Al igual que en el caso del algoritmo genético estándar, los resultados pueden mejorarse utilizando una función de mapeo diferente y utilizando los óptimos reportados por el algoritmo como puntos iniciales en algoritmos numéricos basados en gradiente. A diferencia del algoritmo genético estándar mediante el algoritmo genético de conservación de especies pueden encontrarse más de un óptimo en una sola corrida, pero es necesario determinar el valor adecuado del parámetro de radio. Si no se posee información de la función que se está optimizando como podría ser una distancia aproximada entre dos óptimos o su distribución, inicializar el radio puede ser una tarea complicada que requiera de un análisis por separado o el uso de métodos auxiliares [Li J.-P.02].

El algoritmo de optimización de enjambre de partículas es el que presenta mejores resultados; al igual que el algoritmo genético de conservación de especies esta diseñado para encontrar múltiples óptimos. La desventaja del algoritmo de optimización de enjambre de partículas es que presenta dos parámetros a inicializar, el radio y el número de partículas por especie. Al igual que en algoritmo genético de conservación de especies no hay un método estándar para inicializar el parámetro del radio; una adecuada inicialización depende en gran medida del conocimiento que se tenga de la función a optimizar. El valor del parámetro de número de partículas por especie es de particular importancia, la eficiencia del algoritmo para encontrar más de un óptimo depende de cuantas partículas sean expulsadas de una especie y reinicializadas de forma aleatoria en el espacio de búsqueda. Si son conservadas un gran número de partículas por especie es posible que no queden suficientes para explorar todo es espacio, si se conservan muy pocas, el algoritmo podría tener problemas de convergencia. Al seleccionar el valor del número de partículas a conservar también debe tomarse en cuenta el valor del radio. Un radio muy pequeño implica una gran cantidad de especies, exhibiendo un efecto similar al de seleccionar un número muy grande de partículas a conservar. Un radio muy grande causa un efecto contrario, semejante al de tener muy pocas partículas por especie, mala convergencia. Debe encontrarse una buena relación entre los dos parámetros lo cual no siempre es fácil de lograr.

6.3. Detección de regiones que contienen óptimos

En esta sección se muestran los resultados obtenidos mediante el algoritmo del explorador; se muestran ejemplos con una de las funciones de prueba para algoritmos mul-

timodales. Se muestra además el uso de este algoritmo en la búsqueda de puntos fijos para análisis de sistemas dinámicos.

6.3.1. Máximos de la función de Levy #5

El problema que se analiza es encontrar todos los máximos de la función Levy #5 [Mishra06] descrita por la Ecuación (6.10)

$$\begin{aligned} f(x) &= \sum_{i=1}^{5} i\cos[(i+1)x_1+i]\sum_{i=1}^{5} i\cos[(i+1)x_2+i] \\ &\quad +(x_1+1.42513)^2+(x_2+0.80032)^2 \end{aligned} \tag{6.10}$$

en la región restringida por $x_1 \in [-5,5]$ y $x_2 \in [-5,5]$. La Figura 6.2 muestra la gráfica de la función Levy #5.

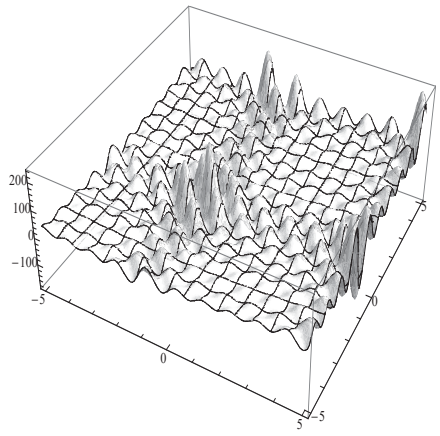


Figura 6.2: Gráfica de la función levy

Para este experimento se decidió utilizar una distribución inicial uniforme de puntos, debido a que como se mostró en el Capítulo 4, genera envolturas convexas regulares y éstos encierran de manera completa la regiones de interés. Después de aplicar el método propuesto para una muestra inicial de $10,000$ puntos necesarios para conseguir una rejilla rectangular uniforme donde la distancia entre puntos seas de 0.1, y de la aplicación de los filtros, todas las 200 regiones con un máximo fueron encontradas. Los resultados son mostrados en la Figura 6.3

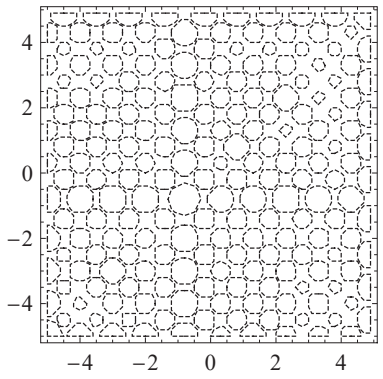


Figura 6.3: Regiones determinadas para la función levy.

El resultado obtenido puede parecer simple, pero si tomamos en cuenta las observaciones de los resultados obtenidos en la sección anterior, un algoritmo genético estándar tendría que ser corrido un mínimo de 200 ocasiones para obtener todos los óptimos; ésto sin considerar que las regiones donde se encuentran los óptimos no son regulares, seleccionar las regiones a descartar puede compararse con la inicialización del parámetro del radio para los algoritmos evolutivos diseñados para encontrar más de un óptimo.

Para los algoritmos que necesitan de un parámetro de radio (conservación de especies, optimización de enjambre de partículas, etc.), si no se tiene información de la función de Levy, ésta presenta regiones con óptimos de diversos tamaños. Encontrar un valor adecuado para el radio puede ser un trabajo complicado.

En lo que respecta al número de individuos en la población inicial, para el caso del algoritmo genético de conservación de especies no hay referencias bibliográficas. Este parámetro tendría que ajustarse mediante experimentación. En el caso de enjambre de partículas, aún sabiendo que tenemos 200 óptimos, y si tenemos un valor adecuado del radio, si el número de partículas por especie es fijado a 20 necesitaríamos $4,000$ partículas; además tenemos que agregar partículas libres para exploración. Tomando el doble de las partículas calculadas, la cantidad necesaria se acerca al número de puntos que se utilizó en el muestreo inicial para el algoritmo del explorador. Además debe tomarse en cuenta de que el algoritmo del explorador calcula todas las regiones en una sola corrida, los algoritmos evolutivos regularmente se corren por al menos 100 generaciones, y se realizan de 10 a 20 corridas, debido a su naturaleza no determinista.

En este caso se ha demostrado que el algoritmo del explorador presenta ventajas frente a los algoritmos evolutivos al encontrar todos los óptimos de la función Levy #5, no necesitar más de los puntos iniciales que se ocuparían para un algoritmo evolutivo y lograr resultados en una sola corrida.

6.3.2. Determinación de puntos fijos en sistemas dinámicos

El algoritmo del explorador fue utilizado para resolver el problema de encontrar todas las soluciones de un sistema de ecuaciones no lineales dentro de una región determinada. El algoritmo propuesto ha sido probado con varios sistemas de ecuaciones, las cuales provienen de varias fuentes y autores [Seydel99, Kuznetsov98, H.-J. Krug85]. Se muestra un ejemplo donde se encuentran todos los puntos fijos para un sistema dinámico.

Un punto fijo de un sistema dinámico puede ser encontrado resolviendo el sistema de ecuaciones diferenciales que modelan el sistema dinámico, cuando todas sus derivadas son iguales a 0. Ésto es, encontrando todas las soluciones de un sistema de ecuaciones no-lineales. El sistema dinámico para nuestro ejemplo es el sistema eléctrico de potencia descrito en la Sección 6.1.

Solo las variables δ_m, δ y V son consideradas, debido a la restricción de la Ecuación (6.1), si hacemos todas las derivadas iguales a 0 esto restringe el valor de ω a 0. Todos los otros símbolos son valores constantes los cuales se indican en la Tabla 6.1.

Los rangos específicos para las variables y el número de puntos que se utilizó para cada una de ellas para la inicialización de la malla son mostradas en la Tabla 6.9.

Tabla 6.9: Valores de los rangos de búsqueda y número de puntos para las variables δ_m, δ, y V.

variable	rango	número de puntos
δ_m	$[0, 0.35]$ rad	36
δ	$[0.05, 0.15]$ rad	16
V	$[1.0, 1.4]$ pu	41

Se utilizó la función descrita en la Sección 5.4.2 para transformar el problema de buscar soluciones en uno de encontrar máximos. La Ecuación (5.14) mapea todos los ceros a uno y los convierte en máximos. Todos los otros valores de f serán mapeados a valores más pequeños de 1. Usando esta transformación el problema es ahora encontrar todos los $\overline{x} \in X$ tales que $g(\overline{x}) = 1$. Hemos transformado el problema de encontrar soluciones en uno

de encontrar máximos.

La primera región encontrada por el algoritmo (ver la Figura 6.4) corresponde a una condición inicial conocida: el punto fijo $(\delta_m, \delta, V) = (0.2858, 0.10662, 1.2295)$ se encuentra dentro de la envoltura convexa generada.

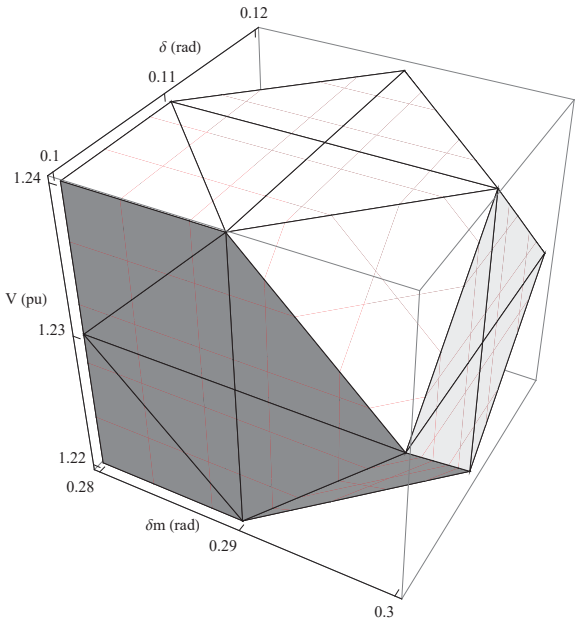


Figura 6.4: Primera envoltura convexa generada.

Como se comentó en la sección anterior, una de las ventajas del método es que la primera región es obtenida en cuestión de algunos segundos. Dado que la función es monotónica dentro de la envoltura convexa, cualquiera de los puntos que forman la envoltura convexa, o puntos interiores a éste pueden ser usados como puntos iniciales de un algoritmo numérico basado en gradiente.

A pesar de ser necesario un mayor número de puntos para el algoritmo del explorador, comparado con los algoritmos evolutivos, las envolturas convexas pueden ser examinadas en cuanto sean generados por el algoritmo, para determinar con precisión cual es el óptimo que está encerrado en ésta (como se explica en el Capítulo 4, tenemos la certeza de que habrá un óptimo encerrado en cada envoltura convexa generada).

6.3.3. Comparación del algoritmo del explorador con el algoritmo de enjambre de partículas con especies

Para mostar una de la principales ventajas que ofrece el algoritmo del explorador, se comparan los resultados obtenidos en un ejemplo del problema de encontrar puntos fijos de un sistema dinámico. El problema se resuelve primero con el algoritmo de optimización de enjambre de partículas con especies y luego se comparan los resultados obtenidos con los encontrados con el algoritmo del explorador.

Dado el sistema dinámico representado por el sistema de ecuaciones no lineales con dos variables y un parámetro

$$\dot{x} = x(x - x^2 - y) \tag{6.11}$$

$$\dot{y} = y\left(x - \frac{1}{u}\right) \tag{6.12}$$

necesitamos encontrar todos los puntos fijos para el sistema dinámico para el valor del parámetro $u = 0.12$, dentro del rango de búsqueda $x \in [0, 4]$ y $y \in [0, 2]$. En ambos casos se utilizó la función de mapeo de la Ecuación (5.14). La función de aptitud se muestra en la Figura 6.5

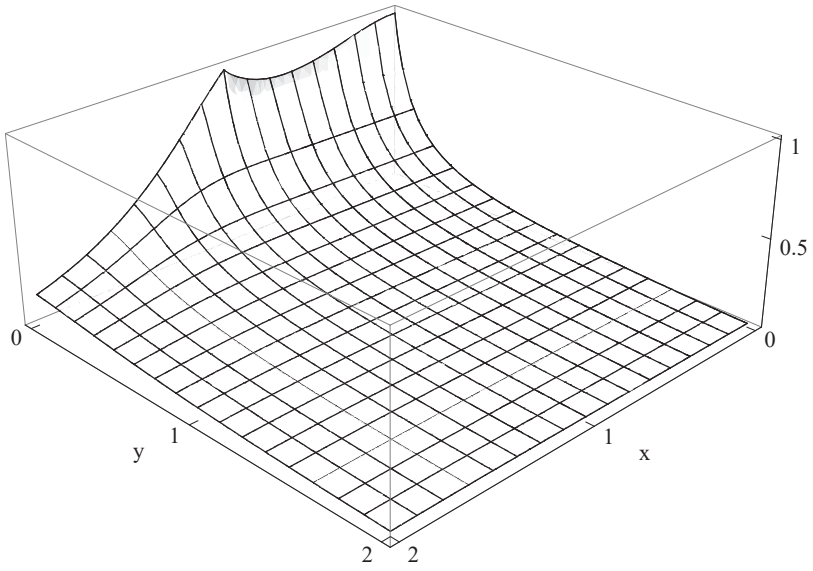


Figura 6.5: Grafica de la función de aptitud para el sistema dinámico descrito por las Ecuaciones (6.11) y (6.12).

Para encontrar los puntos fijos con el algoritmo de optimización de enjambre de partículas, se utilizó un enjambre de 500 partículas, 10 partículas por especie y un radio $r = 0.1$. El algoritmo fue corrido por 100 iteraciones. Después de terminar el ciclo de iteraciones con el algoritmo de optimización de enjambre de partículas se obtuvieron 3 puntos con valores de aptitud superiores a 0.99 los cuales se muestran en la Tabla 6.10

Tabla 6.10: Puntos encontrados mediante el algoritmo de optimización de enjambre de partículas con especies.

coordenadas del punto	valor de aptitud
(1.0, 0.0)	1.0
(0.0, 0.0)	1.0
(0.1011, 0.0)	0.99

Para el algoritmo del explorador se utilizaron los mismos rangos del espacio de búsqueda, también se utilizó la Ecuación (5.14) como función de mapeo y se utilizó una distribución uniforme en forma de rejilla rectangular con una distancia de 0.05 de separación entre puntos consecutivos de una línea, dando un total de $3,200$ puntos en la rejilla. Al correr el algoritmo del explorado sólo se encontraron dos regiones con óptimos, los cuales se muestran en la Figura 6.6.

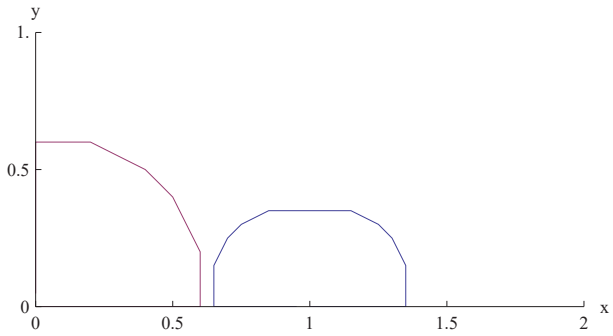


Figura 6.6: Envolturas convexas encontradas por el algoritmo del explorador para el sistema de Ecuaciones (6.11) y (6.12)

Al comparar la Figura 6.6 con los resultados de la Tabla 6.10 puede observarse que el punto $(0.1011, 0.0)$ está dentro de la misma envoltura que el punto $(0, 0)$, dado que en la rejilla la distancia entre puntos consecutivos de una línea es de la mitad del valor del radio en el caso de optimización de enjambre de partículas. Si los dos puntos fueran óptimos se habrían generado dos regiones, una por cada punto.

Dado que se ha generado sólo una región para dos posibles óptimos obtenidos por

el algortimo de optimización de enjambre de partículas, con una frecuencia de barrido más alta, con el algoritmo del explorador, se puede concluir que uno de los óptimos reportados por el algoritmo de optimización de enjambre de partículas es espurio.

Este ejemplo muestra que incluso con un sistema relativamente simple, el algoritmo de optimización de enjambre de partículas puede generar falsos positivos, i.e. puede reportar puntos como óptimos de una función sin que éstos realmente lo sean. Este problema no se presenta en el algoritmo del explorador.

6.4. Trazo de diagramas de bifurcación completos

En esta sección se presentan casos de estudio del trazo de diagramas de bifurcación completos. Para realizar el trazo de diagramas de bifurcación se sigue el planteamiento mostrado en la Sección 5.5, se presentan casos de estudio del trazo de diagrmas de bifurcación con variación de uno y dos parámetros, donde se comparan resultados teoricos y obtenidos en otros trabajos con los resultados obtenidos mediante el método propuesto utilizando el algoritmo de optimización de enjambre de partículas con especies.

6.4.1. Algoritmos numéricos auxiliares

Aunque los algoritmos de inteligencia artificial, en particular los algoritmos evolutivos, proporcionan una forma de realizar la búsqueda automatizada de óptimos para una función objetivo, es posible que sólo se obtengan buenas aproximaciones a éstos, o bien puede presentarse el caso de falsos positivos, individuos que tiene valores altos de aptitud pero que no son óptimos de la función que se esta optimizando. Otro caso se presenta cuando un óptimo esta situado en una región del dominio donde la función objetivo presenta una derivada casi igual a cero en una gran extensión; en particular cuando ésta es mucho mayor al valor del radio. En estos casos se pueden utilizar algoritmos numéricos estándar como auxiliares para mejorar la precisión de los resultados o discriminar entre falsos positivos cuando éstos se encuentran en regiones donde la derivada es muy cercana a cero.

Cabe resaltar que el algoritmo del explorador no tiene el problema de falsos positivos, los óptimos de la función objetivo pueden ser determinados de forma única, además de que las regiones obtenidas se extienden hasta que la condición de descenso estricto sea violada. Esto hace que las regiones se puedan extender y cubrir regiones tan amplias como

la función presente un descenso, lo cual facilita el uso de los algoritmos numéricos dentro de las regiones obtenidas.

6.4.2. Sistemas con un parámetro

Se presentan dos ejemplos: son examinadas las formas normales para sistemas dinámicos con sólo un parámetro que muestran bifurcaciones, los tipos de bifurcación mostrados son: saddle-node, transcrítica, pichfork supercrítica y pichtfork subcrítica. Para cada caso se presenta una comparación de los diagramas de bifurcación generados con el programa xppaut y el algoritmo de optimización de enjambre de partículas para una red eléctrica que ilustra el surgimiento del fenómeno colapso de voltaje cuando se presenta una variación de la carga.

Formas normales para sistemas con un parámetro

Un punto fijo asociado a un sistema dinámico, donde la matriz del jacobiano presenta todos los valores propios con parte real diferente de cero, es llamado un punto fijo hiperbólico. Si examinamos el comportamiento de este punto fijo bajo la variación del parámetro, la condición de que los valores propios de la matriz del jacobiano tengan todos parte real diferente de cero solo puede ser violada en dos formas: la parte real de un valor propio se aproxima a cero, o dos valores propios alcanzan el eje imaginario. Cuando la condición de que todos los valores propios del Jacobiano tengan parte real diferente de cero es violada, las bifurcaciones que presenta el sistema dinámico puede ser una de las bifurcaciones que surgen en los sistemas representados por las Ecuaciones (6.13) a (6.16).

$$\dot{x} = \mu - x^2 \tag{6.13}$$

$$\dot{x} = (\mu - x)x \tag{6.14}$$

$$\dot{x} = (\mu - x^2)x \tag{6.15}$$

$$\dot{x} = (\mu + x^2)x \tag{6.16}$$

Estas ecuaciones son llamadas formas normales o formas normales topológicas, y son empleadas para reducir un sistema no-lineal dado a su forma mas simple posible, preservando la dinámica en una vecindad del punto fijo donde la condición es violada. En este contexto, un sistema dinámico con un parámetro puede ser reescrito mediante

un cambio de variable, como una de las formas normales cerca del punto fijo hiperbólico, teniendo ambos el mismo diagrama de bifurcación. El cambio de variable es regularmente no trivial.

Para las formas normales el algoritmo de optimización de enjambre de partículas fue inicializado con un enjambre de 500 partículas, un máximo de 10 partículas por especie, y un valor de radio $r = 0.15$. El espacio de búsqueda para todas las formas normales fue el intervalo $[-2, 2]$, el valor inicial $\mu = -2.0$ se incrementó en $\Delta\mu = 0.05$ hasta $\mu = 2.0$. Para cada μ_k el algoritmo se corrió 100 iteraciones. Los diagramas de bifurcación que fueron encontrados para cada una de las formas normales son mostradas en la Figura 6.7; estos resultados coinciden con aquellos encontrados en la bibliografía (ver [Seydel99, Kuznetsov98]).

Diagrama de bifurcación para un sistema eléctrico de potencia

Para la comparación de los diagramas de bifurcación creados utilizando algoritmos de inteligencia artificial y los programas estándar, se utilizó la red eléctrica descrita en la Sección 6.1. Todos los parámetros tienen valores fijos durante el análisis, excepto Q_1, la demanda de potencia reactiva de la carga. Los parámetros de la carga, la red y el generador son dados en la Tabla 6.1.

El diagrama de bifurcación del sistema, relacionado la magnitud del voltaje V al parámetro de bifurcación Q_1 (carga de potencia reactiva), es obtenido mediante la aproximación propuesta. Para mostrar la validez del resultado, el mismo análisis es realizado empleando el programa de continuación/bifurcación xppaut.

Para la red eléctrica, el algoritmo de optimización de enjambre de partículas fue inicializado con un enjambre de 500 partículas, 20 partículas por especie y un valor de radio $r = 0.1$. Los intervalos para las cuatro variables para la red eléctrica son mostrados en la Tabla 6.11, el valor inicial $Q_1 = 10.0$ fue incrementado en $\Delta Q_1 = 0.01$ hasta $Q_1 = 11.5$; para cada Q_{1k} el algoritmo de optimización de enjambre de partículas fue corrido por 100 iteraciones. Los puntos fijos que fueron encontrados para la red eléctrica con el algoritmo de optimización de enjambre de partículas y el diagrama de bifurcación generado por el programa xppaut son mostrados en la Figura 6.8.

De la Figura 6.8 puede observarse que se logra obtener un diagrama idéntico al generado por el paquete de software xppaut. Incluso se pudo determinar la región de estabilidad del sistema (marcada con una línea mas gruesa en la Figura 6.8(a) y con puntos

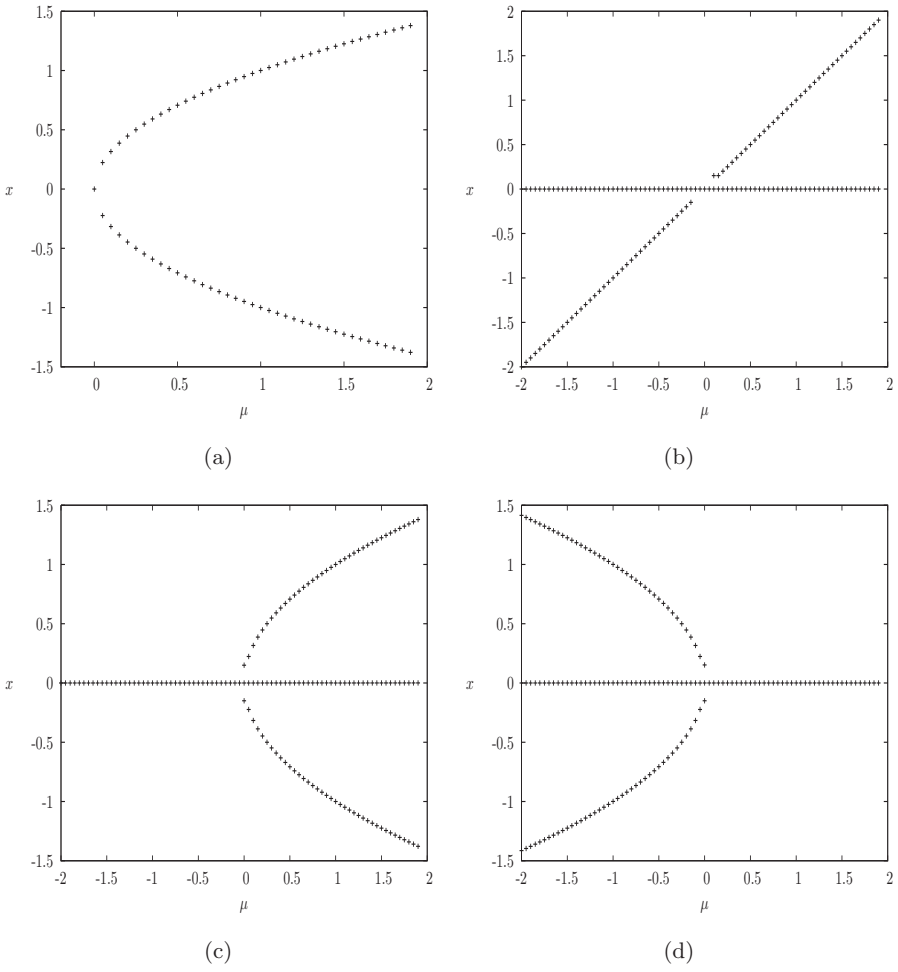
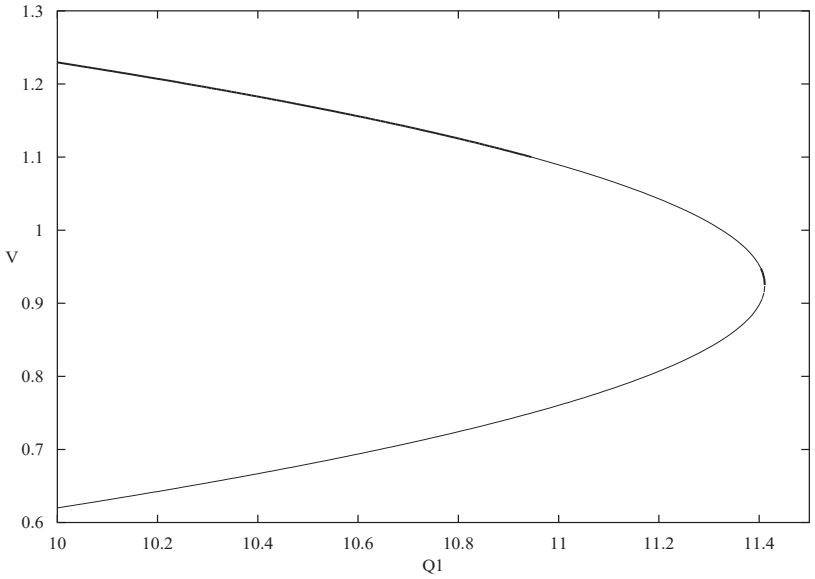


Figura 6.7: Diagramas de bifurcación generados con el algoritmo de optimización de enjambre de partículas para las formas normales: a) bifurcación saddle node (Ecuación (6.13)), b) bifurcación transcrítica (Ecuación (6.14)), bifurcación pichfork supercrítica (Ecuación (6.15)) y bifurcación pichfork subcrítica (Ecuación (6.16))

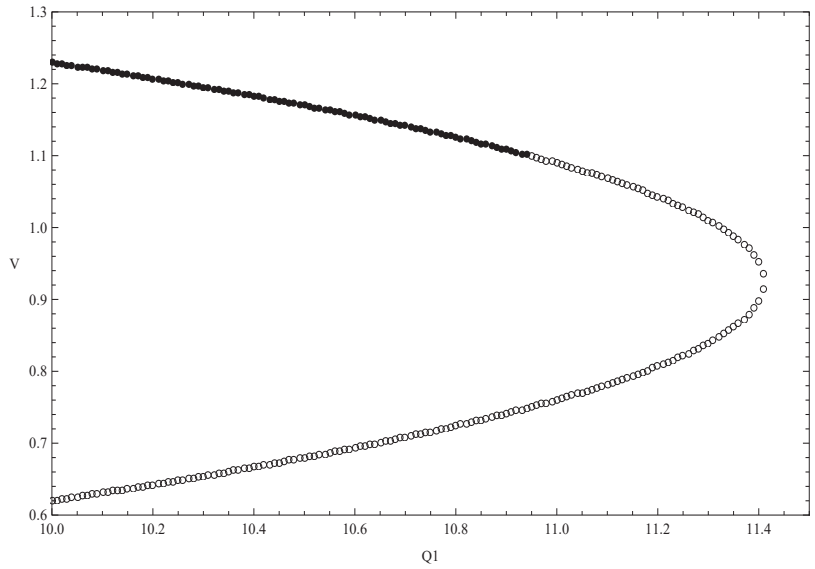
Tabla 6.11: Valores para los rangos de búsqueda de las variables.

variable	rango	variable	rango	variable	rango	variable	rango
δ_m	$[0, 1]$ rad	ω	$[-1, 1]$ rad/seg	δ	$[0, 1]$ rad	V	$[0, 2]$ pu

negros en la Figura 6.8(b)). Utilizando el algoritmo de optimización de enjambre de partículas no es necesario tener un punto inicial estable, ni siquiera es necesario un punto inicial.



(a)



(b)

Figura 6.8: Diagramas de bifurcación obtenidos: a) obtenido con el paquete de software xppaut y b) algoritmo de optimización de enjambre de partículas híbrido.

El método busca todos los puntos fijos (no necesariamente estables) del sistema en la región de búsqueda.

A partir del diagrama generado con el método basado en optimización de enjambre de partículas (Figura 6.8(b)) se realizó el análisis de estabilidad para cada punto calculado; los puntos más obscuros representan puntos estables, mientras que los puntos más claros son inestables. Se observa del diagrama de bifurcación que al rededor del valor del parámetro $Q_1 = 10.96$ el punto fijo se vuelve inestable, y podemos concluir que cerca de este valor para el parámetro $Q_1 = 10.96$ se encuentra un punto de bifurcación. También puede observarse que a partir de un valor $Q_1 = 11.4$ el valor del voltaje V caerá incluso si reducimos el valor del parámetro Q_1, este fenomeno es conocido como colapso de voltaje.

6.4.3. Sistemas con dos parámetros

En diagrama de bifurcación realizado en la sección anterior es realizado variando un sólo parámetro. Para realizar un análisis más completo de un sistema y obtener mayor información de su comportamiento, es necesario realizar diagramas de bifurcación variando más de un parámetro. El análisis con más de un parámetro esta limitado a cambiar un solo parámetro a la vez; esta limitante es impuesta por los algoritmos numéricos existentes [Eusebius97]. Si se requiere, por ejemplo, construir una superficie que represente el diagrama de bifurcación al variar dos parámetros, es necesario realizar el diagrama de bifurcación variando un parámetro y dejando el otro fijo, lo cual da como resultado un conjunto de diagramas de bifurcación. Estos diagramas de bifurcación después son unidos manualmente para reconstruir una superficie [Carvajal-Pérez07].

Con el método establecido en la sección anterior no es necesario fijar un parámetro, cualquier número de parámetros puede ser cambiado al mismo tiempo. Además de que el proceso de generar un diagrama de bifurcación con el método propuesto es realizado sin supervisión; solo deben ser establecidos los parámetros para el método y éste realiza todo los pasos, incluyendo la determinación de la estabilidad de cada uno de los puntos fijos calculados.

En esta sección se genera un diagrama de bifurcación variando dos parámetros. El diagrama de bifurcación obtenido es comparado con resultados obtenidos mediante los métodos estándar. Se analiza el sistema eléctrico de potencia descrito en la Sección 6.1, con la variación de que éste presenta un sistema de excitación rápido en el generador sin

limitador de voltaje de campo. La red eléctrica es mostrada en la Figura 6.9.

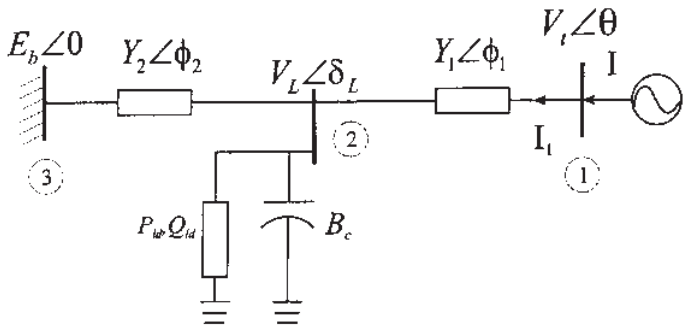


Figura 6.9: Diagrama del sistema eléctrico de potencia.

Tabla 6.12: Valores constantes para los parámetros en el sistema de ecuaciones diferenciales.

parámetro	valor	parámetro	valor	parámetro	valor
P_m	0.4 pu	Y_1	4.9752 pu	Y_2	1.6584 pu
Y	4.9752 pu	ϕ_1	-1.4711 rad	ϕ_2	-1.4711 rad
ϕ	-1.4711 rad	E_b	1.0 pu	E_m	1.0 pu
X_d	1.79 pu	X_q	1.71 pu	X_{dp}	0.169 pu
X_{qp}	0.23 pu	T_{dop}	4.3 seg	T_{qop}	0.85 seg
P_0	0.4 pu	Q_0	0.8 pu	P_1	0.24 pu
Q_1	-0.02 pu	P_2	1.7 pu	Q_2	-1.866 pu
H	2.894 seg	W_b	377.0 rad/seg	D	0.05 pu
P_3	0.2 pu	Q_3	1.6 pu	Q_{1d}	0.0 pu
P_{1d}	0.0 pu	B_c	0.2 pu	K_A	200.0
T_A	0.01 seg	v_{ref}	1.1223 pu		

La dinámica del sistema está gobernada por las seis ecuaciones diferenciales ordinarias 6.17 a 6.22 [Carvajal-Pérez07]

$$\begin{aligned}
\dot{V}_L &= \{P - P_{1d} - P_0 - P_1[(Q - Q_{1d} - Q_0 \\
&\quad -Q_2V_L - (Q_3 - B_c)V_L^2)/Q_1 - P_3V_L]\}/P_2 && (6.17)\\
\dot{\delta}_m &= \omega\omega_b && (6.18)\\
\dot{\omega} &= [-D\omega + P_m - (E_{qp}I_q + E_{dp}I_d + (X_{dp} - X_{qp})I_dI_q)]/2H && (6.19)\\
\dot{E}_{qp} &= \frac{-E_{qp} + (X_d - X_{dp})I_d + E_{fd}}{T_{dop}} && (6.20)\\
\dot{E}_{fd} &= \frac{-E_{fd} + K_A(V_{ref} - V_t)}{T_A} && (6.21)\\
\dot{\delta}_L &= \frac{Q - Q_{1d} - Q_0 - Q_2V_L - (Q_3 - B_c)V_L^2}{Q_1} && (6.22)
\end{aligned}$$

En estas ecuaciones las variables de estado están definidas como sigue: δ_m es la ángulo de fase del generador de voltaje, ω es la velocidad del rotor, δ_L es el ángulo de fase del voltaje de la carga y V_L es la magnitud del voltaje de la carga. Las funciones P y Q que aparecen en estas ecuaciones, representan respectivamente, la potencia activa y reactiva suministrada a la carga por la red. Éstas están definidas en las Ecuaciones (6.23) y (6.24) [Carvajal-Pérez07]

$$\begin{aligned}
P &= V_tV_LY_1\cos(\delta_L - \theta - \phi_1) - V_L^2Y_1\cos\phi_1 \\
&\quad +E_bV_LY_2\cos(\delta_L - \phi_2) - V_L^2Y_2\cos\phi_2 && (6.23)\\
Q &= V_tV_LY_1\sin(\delta_L - \theta - \phi_1) - V_L^2Y_1\sin\phi_1 \\
&\quad +E_bV_LY_2\sin(\delta_L - \phi_2) - V_L^2Y_2\sin\phi_2 && (6.24)
\end{aligned}$$

Las siguientes ecuaciones algebraicas son necesarias para la definición del sistema de ecuaciones diferenciales de las Ecuaciones (6.17) a (6.22) [Carvajal-Pérez07]

$$a = Y_1 V_L \cos(\delta_L - \delta_m - \phi + \phi_1) \quad (6.25)$$

$$b = Y_1 V_L \sin(\delta_L - \delta_m - \phi + \phi_1) \quad (6.26)$$

$$B_p = \sin\phi - Y X_{dp} \quad (6.27)$$

$$A_p = Y X_{qp} - \sin\phi \quad (6.28)$$

$$W_p = \frac{X_{qp} - X_q}{\cos^2\phi - A_p B_p} \quad (6.29)$$

$$E_{dp} = \frac{W_p(Y\cos\phi E_{qp} + bB_p - a\cos\phi)}{1 + W_p Y B_p} \quad (6.30)$$

$$I_q = \frac{\cos\phi(Y E_{qp} - a) - B_p(Y E_{dp} - b))}{-\cos^2\phi + A_p B_p} \quad (6.31)$$

$$I_d = \frac{\cos\phi(Y E_{dp} - b) + B_p(Y E_{qp} - b)}{-\cos^2\phi + A_p B_p} \quad (6.32)$$

$$V_q = E_{qp} + X_{dp} I_d \quad (6.33)$$

$$V_d = E_{dp} - X_{qp} I_q \quad (6.34)$$

$$V_t = \sqrt{V_q^2 + V_d^2} \quad (6.35)$$

$$V_i = \frac{V_d}{V_q} \quad (6.36)$$

$$\theta = \delta_m + \arctan V_i \quad (6.37)$$

Las demás variables que aparecen en las ecuaciones 6.17 a 6.37 son parámetros constantes, relacionados a la carga, a la red y al generador. Todos estos parámetros tienen valores fijos durante el análisis, excepto P_m y T_A, para replicar el diagrama de bifurcación mostrado en la Figura 6.10(a). Los valores para los parámetros están dados en la tabla 6.12.

El diagrama de bifurcación para el sistema, que relaciona la magnitud del voltaje V con los parámetros P_m y T_A es obtenido mediante el método propuesto. El diagrama de bifurcación obtenido es comparado con el diagrama realizado en un trabajo de tesis previo hacer del análisis de sistemas de potencia [Carvajal-Pérez07].

Los valores para el algoritmo de optimización de enjambre de partículas son los siguientes: se utilizó un enjambre de 600 partículas, un número de 25 partículas por especie y un radio $r = 0.25$. Los intervalos de búsqueda para las seis variables consideradas del sistema dinámico son mostradas en la Tabla 6.13. El valor inicial $P_m = 0.4$ fue incrementado en $\Delta P_m = 0.01$ hasta un valor de $P_m = 1.6$; para cada P_{mk} el algoritmo de optimización de enjambre de partículas fue corrido durante 100 ciclos. El valor inicial $T_A = 0.01$ fue

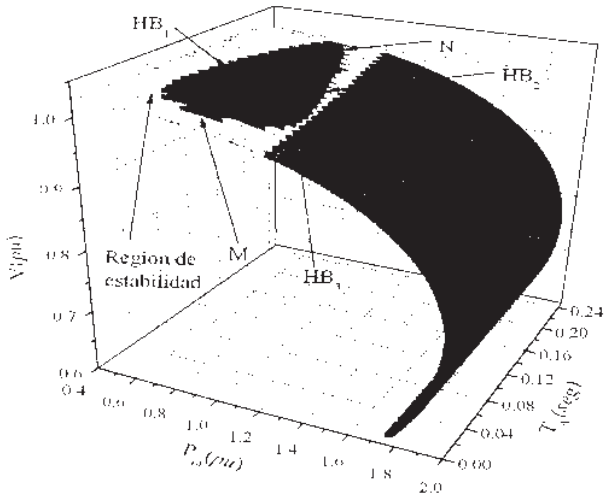
incrementado en $\Delta T_A = 0.0025$ hasta un valor de $T_A = 0.25$. Los diagramas de bifurcación generados con el paquete de software xppaut y el generado con el método propuesto son mostrados en la Figura 6.10.

Tabla 6.13: Rangos de búsqueda para cada una de las seis variables de las Ecuaciones (6.17) a (6.22).

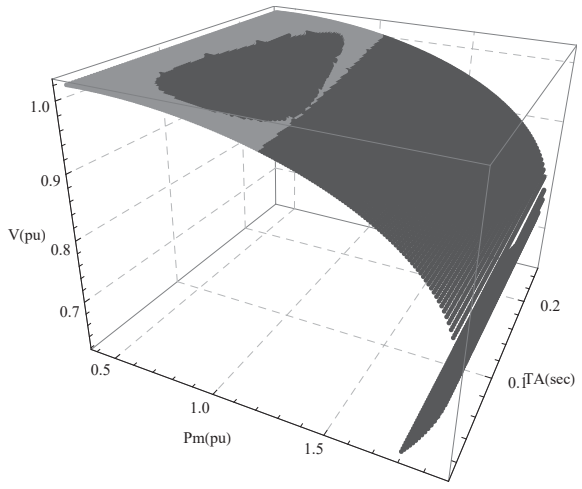
variable	rango	variable	rango
δ_m	$[0.5, 1.1]$ rad	ω	$[-0.3, 0.3]$ rad/seg
δ_L	$[0, 0.3]$ rad	V_L	$[0.6, 1.4]$ pu
E_{fd}	$[2, 2.6]$ pu	E_{qp}	$[0.7, 1.3]$ pu

Como se puede observar en la Figura 6.10 el diagrama de bifurcación obtenido por medio del algoritmo de optimización de enjambre de partículas es el mismo que el generado por el paquete de software xppaut, incluso con mejor resolución. Cabe mencionar que el diagrama generado con xppaut fue creado una línea a la vez, y después cada línea fue unida para formar el gráfico tridimensional. El diagrama generado con el método propuesto fue construido de la misma manera, línea por línea, pero todo el proceso, incluyendo la construcción de la gráfica final fue totalmente automatizado. Otro punto importante a considerar es que el gráfico de la Figura 6.10(a) fue creado en un tiempo de varios meses de trabajo del investigador, el gráfico mostrado en la Figura 6.10(b), en comparación, fue generado en un tiempo total aproximado de dos días, sin supervisión.

En el diagrama generado (Figura 6.10(b)) mediante el método basado en el algoritmo de optimización de enjambre de partículas con especies, es realizado el análisis de estabilidad para cada uno de los puntos. Los puntos estables están representados con un color gris claro y los puntos inestables están representados con un color gris obscuro. Puede observarse que dentro de la zona de estabilidad existe una región inestable; si consideramos una línea para un valor fijo del parámetro $T_A = 0.1$ y variando el parámetro P_m podemos observar tres cambios en la estabilidad del sistema (y por lo tanto tres puntos de bifurcación), pero el número de cambios en la estabilidad es reducido si reducimos el parámetro T_A a valores cercanos a $T_A = 0.02$. De igual manera el número de cambios en la estabilidad es reducido cuando el parámetro T_A es mayor del valor $T_A = 0.22$, esto indica una dependencia de la estabilidad del sistema en el parámetro T_A. Además de la Figura 6.8(b) puede observarse que también se presenta asociado al parámetro P_m el fenomeno de colapso de voltaje.



(a)



(b)

Figura 6.10: Diagramas de bifurcación obtenidos: a) con el paquete de software xppaut y b) con el algoritmo de optimización de partículas híbrido

6.5. Conclusiones del capítulo

En el presente capítulo se han presentado los resultados obtenidos al utilizar los algoritmos de inteligencia artificial, además del método propuesto basado en el algoritmo incremental de envoltura convexa, en la búsqueda de puntos fijos en sistemas dinámicos. Así como el uso de algoritmos evolutivos en el trazo de diagramas de bifurcación. Los algoritmos utilizados para la búsqueda de puntos fijos, para su uso como puntos iniciales, ofrecen una herramienta de búsqueda automatizada eficiente. Además se ha mostrado que en el caso del trazo de diagramas de bifurcación, éstos muestran resultados semejantes a los de los algoritmos estándar con ventajas, como la no dependencia de un punto inicial estable. Para la construcción de diagramas de bifurcación variando más de un parámetro, se pueden obtener resultados sin la necesidad de supervisión.

Capítulo 7

Conclusiones y Trabajo Futuro

En este último capítulo se resumen los resultados obtenidos, y se dan las conclusiones de los mismos. También son presentadas las líneas de investigación que se siguen actualmente y las que se pueden seguir a partir del trabajo realizado.

7.1. Conclusiones

En la presente tesis se ha mostrado el uso de los algoritmos de inteligencia artificial como auxiliares en el proceso de análisis de sistemas dinámicos mediante el trazo de diagramas de bifurcación. Ésto se realiza siguiendo los pasos del análisis de sistemas dinámicos por medio del trazo del diagramas de bifurcación relacionados a uno o varios parámetros, y evaluando el uso de los algoritmos de inteligencia artificial en cada uno de los pasos del análisis.

En la primera parte, la búsqueda de puntos fijos estables, se ha mostrado el desempeño de tres tipos de algoritmos evolutivos; cada uno de ellos presenta características propias en cuanto al número de parámetros a inicializar y el grado de precisión de los puntos fijos encontrados. Aunque la precisión de estos algoritmos no supera a la de los algoritmos numéricos estándar, sí proporcionan una herramienta de búsqueda automatizada. Esta sola característica implica un ahorro de tiempo considerable para el investigador; aún sin tener mucha información de la función que se analiza, es posible encontrar buenas aproximaciones, algunas veces en cuestión de minutos. Aun si la búsqueda de puntos fijos tardara días, se realiza de forma completamente automatizada, sin necesitar la supervisión del investigador. Si los puntos obtenidos no tienen una precisión adecuada, pueden ser usados como

puntos iniciales para los algoritmos numéricos estándar. Otra característica de los algoritmos evolutivos es que no dependen de las propiedades de la función a analizar como son la continuidad y la diferenciabilidad.

Como se comenta en el Capítulo 6 cada uno de los algoritmos evolutivos analizados presenta características propias; aunque el algoritmo de optimización de enjambre de partículas presenta mejores resultados en lo que respecta al valor del error, y es posible encontrar más de un óptimo, éste requiere de más parámetros a inicializar. Con la descripción realizada de las características de los algoritmos evolutivos es posible decidir cual es más adecuado para el problema que se este analizando.

Así pues, se ha demostrado la utilidad de los algoritmos evolutivos en la búsqueda de puntos fijos para el análisis de sistemas dinámicos. También se ha mostrado, dentro del ámbito de los algoritmos evolutivos, no propiamente de su aplicación, sino del desarrollo de los mismo, que su desempeño es afectado por la función de mapeo que se utilice. Existe muy poca información acerca de los efectos de la función de mapeo [Goldberg00] y en la presente tesis sólo se muestra la comparación de los resultados obtenidos con dos diferentes funciones de mapeo.

En esta tesis también se ha presentado un nuevo algoritmo para optimización multimodal, el algoritmo de optimización del explorador. Como se mostró en el Capítulo 3, los algoritmos evolutivos diseñados para problemas multimodales dependen de parámetros adicionales; uno de ellos, el radio, es de particular interés. El valor que sea asignado a este parámetro no sólo afecta el tiempo de corrida del algoritmo, sino que también puede afectar la correcta localización de los óptimos de una función. El problema de encontrar un valor adecuado para el radio, aún es un problema abierto de algoritmos evolutivos. El algoritmo presentado no tiene dependencia en parámetros como el radio.

Como se ha mostrado en los Capítulos 4 y 6, el algoritmo del explorador muestra una mejora considerable al utilizarlo con una distribución uniforme (i.e. una rejilla rectangular) y aunque la complejidad del algoritmo es de $O(N2^N)$, donde N es la dimensión del problema. Si se toma N fija, el algoritmo aún se considera de complejidad polinomial, lo cual implica que es un algoritmo eficiente computacionalmente. Adicionalmente al desarrollo del algoritmo del explorador, también se ha propuesto una representación para facetas en un espacio n-dimensional, que simplifica la implementación del algoritmo de envoltura convexa para el caso general. Esta representación también reduce los recursos de cómputo utilizados por el algoritmo de envoltura convexa al reducir de forma considerable la estructura de

datos de las facetas. Hasta donde el autor conoce no existe ninguna referencia de algoritmos semejantes al que se presenta en esta tesis, el algoritmo del explorador.

En lo que respecta al último paso del análisis de sistemas dinámicos, el trazo de diagramas de bifurcación, se han mostrado resultados para diagramas de bifurcación que dependen de un parámetro, que son comparables cualitativamente con los obtenidos por programas estándar de uso común, con la ventaja de que el método propuesto no depende de un valor inicial. Además, el algoritmo propuesto para el trazo de diagramas de bifurcación, difiere de los algoritmos numéricos estándar en que no se sigue el cambio de un solo punto inicial; son buscados todos los puntos fijos de un sistema en una región dada, sin importar si son inestables, y se siguen los cambios que éstos presentan al variar uno o más parámetros del sistema. El método puede detectar cuando un nuevo punto fijo aparece (o desaparece) al variar un parámetro.

También se presentan resultados para diagramas generados con la variación de dos parámetros. Es aquí donde el método propuesto para el trazo de diagramas de bifurcación, basado en algoritmos evolutivos ha mostrado tener ventajas considerables. Mientras que para realizar un análisis variando dos parámetros con los algoritmos numéricos estándar es necesario generar un conjunto de diagramas con un parámetro fijo y variar el segundo; el método propuesto permite variar dos o más parámetros al mismo tiempo. En lo que respecta al tiempo de cálculo, el diagrama de bifurcación con dos parámetros realizado con los paquetes de software estándar fue realizado en meses de trabajo, mientras que con el método propuesto esto se logra en dos días, y de manera automatizada.

Es necesario notar que los algoritmos evolutivos no dependen de la continuidad de las funciones que se analizan. Esta propiedad puede ser utilizada para analizar sistemas dinámicos que se modelen con funciones discretas, además de sistemas dinámicos híbridos que son representados con funciones discretas, continuas o con funciones de escalón.

7.2. Trabajo futuro

Las líneas de investigación que pueden desprenderse de la presente tesis abarcan diversos campos. Se ha visto que al utilizar una función de mapeo diferente en los algoritmos evolutivos se pueden mejorar los resultados que se obtienen de éstos. Es necesario un análisis más completo para determinar los efectos de diferentes funciones de mapeo, y determinar las características de las funciones que brinden una mejora.

Se pretende extender el método a sistemas de entorno dinámico, y compararlo con el método propuesto para el trazo de diagramas de bifurcación basado en optimización de enjambre de partículas. Al utilizar el algoritmo del explorador se esperan obtener mejores resultados en el trazo de diagramas de bifurcación, ya que este método garantiza la obtención de todos los óptimos de una función, con una frecuencia de muestreo adecuada.

En el trazo de diagramas de bifurcación, los algoritmos evolutivos para entornos dinámicos, como el utilizado para el trazo de diagramas de bifurcación, están en constante desarrollo, debe realizarse un análisis de nuevas variantes del método que permitan tener mejores resultados, tanto en tiempo, como en precisión de los mismos. Además se deben plantear nuevas estrategias para el problema del cálculo del radio; cada subespecie puede variar su radio dependiendo de las propiedades de las subespecies vecinas. Si no hay subespecies cercanas el radio debe aumentarse para cubrir una región más amplia. Si existen subespecies que se intersecten, pueden analizarse sus individuos dominantes para crear una sola subespecie a partir de éstas, o decrementar su radio con el fin de mejorar sus convergencia.

Los experimentos realizados han sido enfocados principalmente al estudio de sistemas dinámicos que modelan sistemas eléctricos, pero el método fue implementado de manera general para cualquier tipo de sistema dinámico. Deben realizarse estudios de sistemas que se presentan en otras áreas de investigación, como son reacciones químicas, sistemas económicos, sistemas biológicos, entre otros.

La presente tesis puede ser usada como base para el análisis de sistemas que presenten características que dificulten su análisis con los métodos numéricos estándar, como pueden ser sistemas que presenten discontinuidades, o tengan regiones donde las funciones no sean diferenciables, así como, sistemas híbridos modelados con funciones tanto continuas como discretas.

Apéndice A

Archivos de ejemplo

En este capítulo se muestran ejemplos del funcionamiento de los algoritmos. Los ejemplos presentados corresponden a los algoritmos de optimización de enjambre de partículas y el algoritmo del explorador.

A.1. Optimización de enjambre de partículas

Listado del archivo sistema-biologico.lisp

```
;; ---------------------------------------------------------------------
;; PARAMETROS
;; ---------------------------------------------------------------------

(defvar gca 0.1d0)
(defvar i-local 0.0d0)

;; ---------------------------------------------------------------------
;; CONSTANTES
;; ---------------------------------------------------------------------

(defparameter vl -60.0d0)
(defparameter vca 120.0d0)
(defparameter gl 2.0d0)
```

```
(defparameter c 20.0d0)
(defparameter v1 -1.2d0)
(defparameter v2 18.0d0)

;; ---------------------------------------------------------------------
;; GENERICOS
;; ---------------------------------------------------------------------

(defgeneric sistema-biologico (variables))
(defgeneric evaluacion-sistema-biologico (variables))

;; ---------------------------------------------------------------------
;; METODO sistema-biologico
;; ---------------------------------------------------------------------

(defmethod sistema-biologico ((variables list))
  (let ((v (nth 0 variables))
        (ica 0.0d0)
        (minf 0.0d0)
        (v-res 0.0d0))
    (setf minf (* 0.5d0 (+ 1.0 (tanh (/ (- v v1) v2)))))
    (setf ica (* gca minf (- v vca)))
    (setf v-res (/ (- (+ i-local (* gl (- vl v))) ica) c))
    (list v-res)))

;; ---------------------------------------------------------------------
;; METODO evaluacion-sistema-biologico
;; ---------------------------------------------------------------------

(defmethod evaluacion-sistema-biologico ((variables list))
  (/ 1.0d0 (+ 1.0d0 (norma (sistema-biologico variables)))))
```

Listado del archivo sabana-sistema-biologico.lisp

```
(load "pso")
(load "sistema-biologico")

(setf *random-state* (make-random-state t))

(defconstant rangos-sistema '((-200.0d0 100.0d0)))

(defvar enjambre1)
(defvar semillas)

(setf enjambre1 (make-instance 'enjambre :c1 2.1d0 :c2 2.1d0))
(setf gca 0.1d0)

(dotimes (k 100)
  (setf (enjambre-particulas enjambre1)
        (generar-enjambre rangos-sistema 100 'evaluacion-nueve-tres))

  (dotimes (j 101)
    (dotimes (i 100)
      (calcular-velocidades-especies enjambre1 1.5d0 10 rangos-sistema
                                     'evaluacion-sistema-biologico)
      (actualizar-posiciones enjambre1 'evaluacion-sistema-biologico))
    (setf semillas (obtener-semillas enjambre1 1.5d0 10))
    (format t "(~S ~S (~%" gca i-local)
    (dolist (semilla semillas)
      (format t "~S~%" (particula-posicion semilla)))
    (format t "))~%")
    (incf i-local 4.25d0))
  (incf gca 0.1d0))

(quit)
```

Ejemplo de salida sabana-sistema-biologico.lisp

```
(0.1d0 0.0d0 (
(103.42828797816618d0)
(99.54781460886315d0)
(97.30688648092875d0)
(94.64674929449922d0)
(-195.9441349511508d0)
(-194.307979172522d0)
(76.30247607000453d0)
(-191.04673581422855d0)
(72.99819251908399d0)
(-187.34335329793655d0)
(68.47374764704881d0)
(-11.12100716032463d0)
(-105.54536576070677d0)
(-15.433921980082511d0)
(-95.57668130630171d0)
(-25.02685079214829d0)
(-27.884479382918474d0)
(-90.76470904340678d0)
(-88.4848063531388d0)
(-80.67673358945058d0)
(-78.90263234426584d0)
(-76.85355760770646d0)
(-46.41732074936718d0)
(-48.892012384352824d0)
(-69.0284477132073 7d0)
(-51.66135600459612d0)
(-66.69183778155349d0)
(-53.887552206090234d0)
(-59.182077977506644d0)
))
```

Listado del archivo sistema-biologico-dos.lisp

```
;; ---------------------------------------------------------------------
;; PARAMETROS
;; ---------------------------------------------------------------------

(defvar u 2.0d0)

;; ---------------------------------------------------------------------
;; GENERICOS
;; ---------------------------------------------------------------------

(defgeneric sistema-biologico-dos (variables))
(defgeneric evaluacion-sistema-biologico (variables))

;; ---------------------------------------------------------------------
;; METODO sistema-biologico-dos
;; ---------------------------------------------------------------------

(defmethod sistema-biologico-dos ((variables list))
  (let ((xl (nth 0 variables))
        (yl (nth 1 variables))
        (x1 0.0d0)
        (y1 0.0d0))
    (setf x1 (* xl (- xl (expt xl 2) yl)))
    (setf y1 (* yl (- xl (/ 1.0d0 u))))
    (list x1 y1)))

;; ---------------------------------------------------------------------
;; METODO evaluacion-sistema-biologico
;; ---------------------------------------------------------------------

(defmethod evaluacion-sistema-biologico ((variables list))
```

```
  (/ 1.0d0 (+ 1.0d0 (norma (sistema-biologico-dos variables)))))
```

Listado del archivo envoltura-sistema-biologico.lisp

```
(load "malla-clase")
(load "sistema-biologico-dos")

(defvar malla1 (make-instance 'malla))

(generar-malla malla1 ' ((0.0d0 2.0d0) (0.0d0 4.0d0)) '(41 81))
(evaluar-malla malla1 'evaluacion-sistema-biologico)
(ordenar malla1 'mayor-que)
(calcular-convex-hull malla1 'evaluacion-sistema-biologico)

(quit)
```

Ejemplo de salida envoltura-sistema-biologico.lisp

```
1.0 0.0 1.0
1.05 0.0 0.9477550053311218
1.1 0.0 0.8920606601248883
1.1500000000000001 0.0 0.8344633357671846
1.2000000000000002 0.0 0.7763975155279501
1.25 0.0 0.7191011235955056
1.3 0.0 0.6635700066357
1.35 0.0 0.6105472029306265
1.35 0.05 0.5595767485870685
1.35 0.1 0.4898033153065603
1.35 0.15000000000000002 0.4268142721934358
1.3 0.25 0.3395308610330084
1.25 0.30000000000000004 0.3068691505836673
1.1500000000000001 0.35000000000000003 0.2789421380042646
1.1 0.35000000000000003 0.27919405198344643
```

```
1.05 0.35000000000000003 0.27902077080485366
1.0 0.35000000000000003 0.2785189176773506
0.9500000000000001 0.35000000000000003 0.2777703328439812
0.9 0.35000000000000003 0.27684199612835736
0.8500000000000001 0.35000000000000003 0.2757870574758525
0.75 0.30000000000000004 0.3051977519290177
0.7000000000000001 0.25 0.3438152592693576
0.65 0.15000000000000002 0.4643386946070877
0.65 0.1 0.5640826057725742
0.65 0.05 0.7137159082781386
0.65 0.0 0.8711749972775782
0.7000000000000001 0.0 0.8718395815170008
0.75 0.0 0.8767123287671232
0.8 0.0 0.8865248226950355
0.8500000000000001 0.0 0.9022217209879329
0.9 0.0 0.9250693802035153
0.9500000000000001 0.0 0.9568233464896543
```

Referencias

[Abed84] Abed, E. y Varaiya, P. Nonlinear oscillations in power systems. *International Journal of Electric Power and Energy Systems*, 6(1):37–43, January 1984.

[Abed92] Abed, E., Alexander, J., Wang, H., Hamdan, A., y Lee, H. Dynamic bifurcations in a power system model exhibiting voltage collapse. *Tech. Research Report, Systems Research Center*, February 1992.

[Aggarwal00] Aggarwal, V. Solving transcendental equations using genetic algorithms, 2000.
URL `http://web.mit.edu/varun_ag/www/techwrite.html`

[Ajjarapu92] Ajjarapu, V. y Lee, B. The application of bifurcation theory to study the nonlinear dynamical phenomena in an electrical power system. *IEEE Trans. on Power Systems*, 7(1):424–432, February 1992.

[Apostol74] Apostol, T. M. *Mathematical Analysis*. Addison-Wesley, Massachusetts, 1974.

[Baker87] Baker, J. E. Reducing bias and inefficiency in the selection algorithm. *in Proceedings of the Second International Conference on Genetic Algorithms and their Application (Hillsdale)*, págs. 14–21, 1987.

[Carvajal-Pérez07] Carvajal-Pérez, H. R. *Análisis Multiparamétrico de Bifurcaciones en Sistemas Eléctricos*. Proyecto Fin de Carrera, Universidad Mi-

choacana de San Nicolás de Hidalgo, Morelia, Michoacán, Agosto 2007.

[Cañizares95] Cañizares, C. A. On bifurcations, voltage collapse and load modeling. *IEEE Trans. on Power Systems*, 10(1):512–522, February 1995.

[Chiang94] Chiang, H., Connen, T., y Flueck, A. Bifurcation and chaos in electric power systems. *Journal of the Franklin Institute*, 331B(6):1001–1036, 1994.

[Clerc02] Clerc, M. y Kennedy, J. The particle swarm: explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, 2002.

[Coello99] Coello, C. A. C. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems*, 1(3):129–156, 1999.
URL `citeseer.ist.psu.edu/coello98comprehensive.html`

[Crina Grosan06] Crina Grosan, A. A. y Gelbukh, A. Evolutionary method for nonlinear systems of equations. *Advances in Artificial Intelligence*, págs. 283–293, 2006.

[D.J.95] D.J., H. Special issue on nonlinear phenomena in power systems. *Proceedings of IEEE*, 83(11), November 1995.

[Doedel97] Doedel, E., Champneys, A., Fairgrieve, T., Kuznetsov, Y., Sandstede, B., y Wang, X. Auto97 : Continuation and bifurcation software for ordinary differential equations; available by ftp from ftp, 1997.
URL `citeseer.ist.psu.edu/219163.html`

[Ermentrout06] Ermentrout, B. Xppaut5.96 - the differential equations tool, 2006.
URL `http://www.pit.edu/ phase`

[Eusebius97] Eusebius, D. "auto", software for continuation and bifurcation problems in ordinary differential equations. 1997.

[Franco P. Preparata85] Franco P. Preparata, M. I. S. *Computational Geometry an Introduction*. Springer-Verlag, New York, 1985.

[Goldberg00] Goldberg, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, London, 2000.

[Govaerts00] Govaerts, W. J. *Numerical Methods for Bifurcations of Dynamic Equilibria*. Society for Industrial and Applied Mathematics, Philadelphia, 2000.

[H.-J. Krug85] H.-J. Krug, L. K. Ein oszillierendes modellsystem mit autokatalytischem teilschritt. *Z. Phys. Chemie*, 266:65–73, 1985.

[Ian Dobson88] Ian Dobson, J. S. T., Hsiao Dong Chiang. A model of voltage collapse in electric power systems. *IEEE proceedings of 27th. Conference on Decision and Control*, págs. 2104–2109, December 1988.

[K.86] K., W. Modeling of power system components at severe disturbances. *CIGRÉ paper 38-18, International conference on large high voltage electric systems*, August 1986.

[Kennedy95] Kennedy, J. y Eberhart, R. C. Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks*, págs. 1942–1948, 1995.

[Kennedy01] Kennedy, J., Eberhart, R. C., y Shi, Y. *Swarm intelligence*. Morgan Kaufmann, San Francisco, 2001.

[Kincaid91] Kincaid, D. y Cheney, W. *Numerical Analysis Mathematics of Scientific Computing*. Books/Cole, Pacific Grove, California, 1991.

[Kuznetsov98] Kuznetsov, Y. A. *Elements of applied bifurcation theory*. Springer-Verlag, New York, 1998.

[Kwanty86] Kwanty, H. G., Pasrija, A. K., y Bahar, L. Y. Static bifurcations in electric power networks: Loss of steady-state stability and voltage collapse. *IEEE Trans. on Circuits and Systems*, CAS-33(10):981–991, October 1986.

[Levenberg44] Levenberg, K. A method for the solution of certain problems in least squares. *Quart. Appl. Math.*, 2:164–168, 1944.

[Li J.-P.02] Li J.-P., P. G., Balazs M. y P., C. A species conserving genetic algorithm for multimodal function optimization. *Evolutionary Computation*, págs. 207–234, 2002.

[Li06] Li, X., Branke, J., y Blackwell, T. Particle swarm with speciation and adaptation in a dynamic environment. *En GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, págs. 51–58. ACM Press, New York, NY, USA, 2006. ISBN 1-59593-186-4. doi: http://doi.acm.org/10.1145/1143997.1144005.

[Mahfoud95] Mahfoud, S. W. *Niching methods for genetic algorithms.* Tesis Doctoral, Urbana, IL, USA, 1995.
URL `citeseer.ist.psu.edu/mahfoud95niching.html`

[Marquardt63] Marquardt, D. An algorithm for least-squares estimation of nonlinear parameters. *SIAM J. Appl. Math.*, 11:431–441, 1963.

[Miller96] Miller, B. L. y Shaw, M. J. Genetic algorithms with dynamic niche sharing for multimodal function optimization. *En International Conference on Evolutionary Computation*, págs. 786–791. 1996.
URL `citeseer.ist.psu.edu/miller96genetic.html`

[Mishra06] Mishra, S. K. Performance of repulsive particle swarm method in global optimization of some important test functions: A fortran program. *Munich Personal RePEc Archive*, August 2006.
URL `http://ssrn.com/abstract=924339`

[Nocedal99] Nocedal, J. y Wright, S. J. *Numerical Optimization.* Springer-Verlag, New York, 1999.

[Nyquist28] Nyquist, H. Certain topics in telegraph transmission theory. *Transactions AIEE*, 47:617–644, April 1928.

[O'Rourke98] O'Rourke, J. *Computational Geometry in C*. Cambridge University Press, New York, 1998.

[Parrott06] Parrott, D. y Li, X. Locating and tracking multiple dynamic optima by a particle swarm model using speciation. *En IEEE Transactions on Evolutionary Computation*, tomo 10, págs. 440–458. 2006.

[Petrowski97] Petrowski, A. A new selection operator dedicated to speciation. *En* T. Bäck, ed., *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA97)*. Morgan Kaufmann, San Francisco, CA, 1997.
URL `citeseer.ist.psu.edu/petrowski97new.html`

[Randy L. Haupt04] Randy L. Haupt, S. E. H. *Practical Genetic Algorithms*. John Wiley & Sons, New Jersey, 2004.

[Richter83] Richter, S. L. y DeCarlo, R. A. Continuation methods: Theory and applications. *IEEE Transactions on Circuits and Systems*, CAS-30(6):347–352, June 1983.

[Rudin76] Rudin, W. *Principles of Mathematical Analysis*. McGraw-Hill, New York, 1976.

[Sacks91] Sacks, E. P. Automatic analysis of one-parameter planar ordinary differential equations by intelligent numeric simulation. *Artif. Intell.*, 48(1):27–56, 1991. ISSN 0004-3702. doi: http://dx.doi.org/10.1016/0004-3702(91)90079-Y.

[Seydel94] Seydel, R. *Practical Bifurcation and Stability Analysis From Equilibrium to Chaos*. Springer-Verlag, 2ª ed., 1994.

[Seydel99] Seydel, R. World of bifurcation. online collection and tutorials of nonlinear phenomena, 1999.
URL `http://www.bifurcation.de/`

[Shannon49] Shannon, C. E. Communication in the presence of noise. *Proc. Institute of Radio Engineers*, 37(1):10–21, January 1949.

[Singh06] Singh, G. y Deb, K. Comparison of multi-modal optimization algorithms based on evolutionary alorithms. *Genetic And Evolutionary Computation Conference*, págs. 1305–1312, 2006.

[Strogatz00] Strogatz, S. H. *Strogatz S. Nonlinear dynamics and chaos: with applications to physics, biology, chemistry and engineering.* Perseus books, 2000.

[William H. Press92] William H. Press, W. T. V., Saul A. Teukolsky y Flannery, B. P. *Numerica Recipies in C, The Art of Scientific Computing.* Cambridge University Press, Cambridge, 1992.