INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

AJUSTE DE CURVAS CON MÚLTIPLES CORRELACIONES USANDO PROGRAMACIÓN GENÉTICA

T E S I S

QUE PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA
ING. ÁNGEL HERNÁNDEZ CASTAÑEDA

DIRECTOR DE TESIS
DR. FRANCISCO HIRAM CALVO CASTRO

México, D. F., diciembre de 2013

A Dios,
a mis padres
y
a mis
hermanos

RESUMEN

El problema de encontrar un modelo matemático para variables observadas empíricamente puede considerarse como la búsqueda en un espacio de posibles soluciones para un programa que produce la salida deseada dada una entrada particular.

Esta salida deseada se puede encontrar a través del paradigma de la programación genética, que se alimenta de una población (conjunto de soluciones) de programas en una competencia Darwiniana mediante operaciones genéticas que reconocen la aptitud de cada miembro de la población después de la reproducción y mutación. En la presente investigación buscamos funciones matemáticas (mediante un sistema de programación genética) compuestas por variables de entrada (series de datos) que representan información del área de la economía. El propósito es construir el modelo matemático de cierto fenómeno económico donde las variables de entrada o variables independientes son otros fenómenos económicos que el sistema de programación genética puede incluir en la solución. El objetivo de crear un modelo matemático, que esté formado por otros fenómenos económicos, es de analizarlo y encontrar las variables relevantes que lo componen. De esta forma identificamos si el fenómeno económico estudiado está relacionado con algún otro fenómeno.

Al implementar un sistema de programación genética que genere modelos matemáticos compuestos de múltiples variables para su posterior análisis, ampliamos la utilidad de estos sistemas para poder usarlos en el análisis de fenómenos económicos, con el fin de identificar relaciones que no se reconocen fácilmente al observar un conjunto de datos.

ABSTRACT

The problem of finding a mathematical model for empirically observed variables can be regarded as searching in a space of possible solutions for a program which produces the desired output given a particular input.

This desired output can be found through the paradigm of genetic programming, which is fed of a population (solution set) of programs in a Darwinian competition by means of genetic operations recognizing fitness after reproduction and mutation. In this research we seek mathematical functions (using a genetic programming system) composed of input variables (data sets) representing economic information. The purpose is to build a mathematical model of some economic phenomenon where the input variables or independent variables are other economic phenomena that the genetic programming system can be included in the solution. The goal of creating a mathematical model that is composed of other economic phenomena is to analyze and find the relevant component variables. Thus, we identify whether the economic phenomenon under study is related to other economic phenomenon.

By implementing a genetic programming system to generate mathematical models compounds of multiple variables for further analysis, we extend the utility of genetic programming systems to use in the analysis of economic phenomena in order to identify relationships that are not easily recognized by observing a data set.

Agradecimientos

A Dios, por darme salud, bienestar y por apoyarme en momentos difíciles.

A mis padres: Lucia Castañeda Martínez y Felipe Hernández Fonseca, por enseñarme a valorar la vida, por el apoyo incondicional, por motivarme a no renunciar jamás a mis sueños, por ayudarme a tomar las decisiones correctas.

A mis hermanos: Néstor, Iago Emmanuel, Brisa Faridi, Felipe Jeshua y Brenton Eduard, por el toque de alegría que le dan a mi vida.

A mi abuela Andrea Guadalupe, por el cariño y cuidado que me dio.

A mi novia Clau y a mi tía Rey, por el apoyo moral.

A mis abuelos, Teresa, Melchor, Raymundo por regalarme un poco de su sabiduría.

A mis tíos, Chuy, Santa, Adela, Elías, Esteban Castañeda, Tirzo, por darme ánimos para seguir adelante.

A mis padrinos Luis y Jorge, por el apoyo brindado.

A mi asesor, el Dr. Francisco Hiram Calvo Castro, por guiar nuestra investigación con entusiasmo y dedicación.

A los Doctores, Salvador Godoy, Edgardo Manuel Felipe Riverón, Germán Téllez, Sergio Suárez, Alexander Gelbukh, por brindarme un poco de su valioso tiempo.

A mis amigos Marco, Javier, Erick, Edgar, Julio, por los buenos momentos que pasamos.

Al Centro de Investigación en Computación y al CONACYT por el conocimiento y apoyo que me brindaron.

ÍNDICE

ÍNDICE I

ÍNDICE DE TABLAS III

ÍNDICE DE FIGURAS V

1. CAPÍTULO I - INTRODUCCIÓN 1

1.1. Generalidades sobre el tema 2

1.2. Motivación 4

1.3. Descripción de problema 4

1.4. Objetivos 5

1.4.1. Objetivo general 5

1.4.2. Objetivos específicos 5

1.5. Aportaciones de la tesis 5

1.6. Organización de la tesis 6

2. CAPÍTULO II - ESTADO DEL ARTE 8

2.1. Ajuste de curvas con métodos determinísticos 9

2.1.1. Herramienta de ajuste de curvas de MATLAB 9

2.1.2. Ajuste de curvas con Microsoft EXCEL 11

2.1.3. Ajuste de curvas con LabFit 11

2.2. Ajuste de curvas con heurísticas 12

2.2.1. Aplicación de algoritmos genéticos en ajuste de curvas 12

2.2.2. Algoritmos genéticos, un enfoque para el ajuste de curvas 14

2.2.3. Resolución de ajuste de curvas con programación genética 20

2.2.4. Ajuste de curvas usando algoritmos genéticos 22

2.3. Conclusiones 29

3. CAPÍTULO III - PROGRAMACIÓN GENÉTICA Y AJUSTE DE CURVAS 31

3.1. Introducción a la programación genética 32

3.1.1. Breve historia de la Programación Genética 34

3.1.2. Representación 35

3.1.3. Inicialización de la población 36

3.1.4. Selección de la población 38

3.1.5. Recombinación y mutación 42
3.1.6. Especificaciones importantes de la programación genética 44
3.1.7. La evolución con estructuras diferentes 53
3.2. Conceptos básicos de ajuste de curvas con polinomios 54
3.2.1. Ajuste de curvas por mínimos cuadrados 55
3.3. Ajuste de curvas mediante programación genética 56
4. CAPÍTULO IV. IMPLEMENTACIÓN DE "GP Curve Fitting" 58
4.1. GP Curve Fitting 59
4.2. El conjunto primitivo 59
4.3. La medida de aptitud 60
4.4. Elitismo 61
4.5. Parámetros del algoritmo 62
4.6. Operadores de recombinación y mutación 62
4.7. Criterio de término de ejecución 64
5. CAPÍTULO V – RESULTADOS OBTENIDOS 65
5.1. Estudios previos 66
5.2. Sobre los datos usados para los experimentos 67
5.3. Experimento 1. Ajuste del índice de precios de materias primas agrícolas 70
5.4. Experimento 2. Ajuste del índice de precios de alimentos 73
5.5. Experimento 3 - Ajuste del índice de precios de bebidas 75
5.6. Experimento 4 - Ajuste del índice de precios de combustibles 77
5.7. Experimento 5 - Ajuste del índice de precios de insumos industriales 79
5.8. Experimento 6 – Ajuste del índice de precios de alimentos y bebidas 81
5.9. Experimento 7 – Ajuste del índice de precios de petróleo crudo 83
5.10. Experimento 8 – Ajuste del índice de precios de no combustibles 85
5.11. Experimento 9 – Ajuste del índice de precios general de mercancías 88
5.12. Comparación de resultados 90
6. CAPÍTULO 6 – CONCLUSIONES Y TRABAJO FUTURO 96
6.1 Conclusiones 97
6.2 Sobre usos del sistema de programación genética "GP Curve Fitting" 98
6.3 Sobre el trabajo futuro 99
Referencias 100

ÍNDICE DE TABLAS

Tabla 2.1: Ejemplo de recombinación intermedia ampliada con $\alpha = 0.5$ 16

Tabla 2.2: Coeficientes encontrados por el algoritmo genético y coeficientes encontrados en el experimento de Karr et al. (1991)..... 18

Tabla 2.3: Coeficientes encontrados después de 250 generaciones. Comparación de valores verdaderos y valores encontrados por el algoritmo genético (AG)..... 19

Tabla 2.4: Coeficientes encontrados después de 2500 generaciones por el algoritmo genético 20

Tabla 2.5: Datos experimentales para el ajuste de curvas 22

Tabla 2.6: Coeficientes encontrados por el algoritmo genético en 50 generaciones. (evaluado en 100 generaciones)*..... 27

Tabla 2.7: Coeficientes encontrados por el algoritmo genético en 300 generaciones 28

Tabla 2.8: Coeficientes encontrados por el algoritmo genético en 500 generaciones 28

Tabla 3.1: Ejemplo de conjunto primitivo (conjunto de funciones y conjunto de terminales). .. 46

Tabla 5.1: Experimento de Kamal y Eassa (2002) 66

Tabla 5.2: Mercancías agrupadas por categoría (a)..... 67

Tabla 5.3: Mercancías agrupadas por categoría (b)..... 68

Tabla 5.4: Datos obtenidos del ajuste del índice de precios de materias primas agrícolas, donde "y" representa los datos experimentales y "f(x)" la función generada 72

Tabla 5.5: Análisis de la función obtenida en el ajuste del índice de precios de materias primas agrícolas 72

Tabla 5.6: Datos obtenidos del ajuste del índice de precios de alimentos, donde "y" representa los datos experimentales y "f(x)" la función generada..... 74

Tabla 5.7: Análisis de la función obtenida en el ajuste del índice de precios de alimentos..... 75

Tabla 5.8: Análisis de la función obtenida en el ajuste del índice de precios de bebidas 76

Tabla 5.9: Datos obtenidos del ajuste del índice de precios de bebidas, donde "y" representa los datos experimentales y "f(x)" la función generada 77

Tabla 5.10: Análisis de la función obtenida en el ajuste del índice de precios de combustibles 78

Tabla 5.11: Datos obtenidos del ajuste del índice de precios de combustibles, donde "y" representa los datos experimentales y "f(x)" la función generada 79

Tabla 5.12: Análisis de la función obtenida en el ajuste del índice de precios de insumos industriales..... 80

Tabla 5.13: Datos obtenidos del ajuste del índice de precios de insumos industriales, donde “y” representa los datos experimentales y “f(x)” la función generada 81

Tabla 5.14: Datos obtenidos del ajuste del índice de precios de alimentos y bebidas, donde “y” representa los datos experimentales y “f(x)” la función generada 82

Tabla 5.15: Análisis de la función obtenida en el ajuste del índice de precios de alimentos y bebidas..... 82

Tabla 5.16: Datos obtenidos del ajuste del índice de precios de petróleo crudo, donde “y” representa los datos experimentales y “f(x)” la función generada. 84

Tabla 5.17: Análisis de la función obtenida en el ajuste del índice de precios de petróleo crudo 84

Tabla 5.18: Datos obtenidos del ajuste del índice de precios de no combustibles, donde “y” representa los datos experimentales y “f(x)” la función generada 86

Tabla 5.19: Análisis de la función obtenida en el ajuste del índice de precios de no combustibles 87

Tabla 5.20: Datos obtenidos del ajuste del índice de precios general de mercancías, donde “y” representa los datos experimentales y “f(x)” la función generada 89

Tabla 5.21: Análisis de la función obtenida en el ajuste del índice de precios general de mercancías 89

Tabla 5.22: Comparación de resultados entre MATLAB y GP Curve Fitting. El error cuadrático medio de GP Curve Fitting es obtenido del promedio de la ejecución de 30 experimentos 95

ÍNDICE DE FIGURAS

Figura 2.1: Curva estimada y curva original. 13
Figura 2.2: Representación gráfica de datos experimentales para el ajuste de curvas 21
Figura 3.1: Esquema gráfico de la programación genética 32
Figura 3.2: Estructura de árbol representando el programa (/ (+ x y) (- z 4)) 36
Figura 3.3: Pasos para construir un árbol de profundidad 2, usando el método completo 37
Figura 3.4: Árbol de profundidad 2, construido con el método expansivo 38
Figura 3.5: Aplicación de la recombinación. (pc = punto de cruce) 42
Figura 3.6: Aplicación de mutación por subárbol. (pc = punto de cruce) 43
Figura 3.7: Ejemplo de interpretación de la sintaxis de un árbol 51
Figura 5.1: Gráfica del ajuste del índice de precios de materias primas agrícolas 71
Figura 5.2: Gráfica del ajuste del índice de precios de alimentos 73
Figura 5.3: Gráfica del ajuste del índice de precios de bebidas 76
Figura 5.4: Gráfica del ajuste del índice de precios de combustibles 78
Figura 5.5: Gráfica del ajuste del índice de precios de insumos industriales 80
Figura 5.6: Gráfica del ajuste del índice de precios de alimentos y bebidas 83
Figura 5.7: Gráfica del ajuste del índice de precios del petróleo crudo 85
Figura 5.8: Gráfica del ajuste del índice de precios de no combustibles 87
Figura 5.9: Gráfica del ajuste del índice de precios general de mercancías 88
Figura 5.10: Ajuste del índice de precios de materias primas agrícolas generado por MATLAB 90
Figura 5.11: Ajuste del índice de precios de alimentos generado por MATLAB 91
Figura 5.12: Ajuste del índice de precios de bebidas generado por MATLAB 91
Figura 5.13: Ajuste del índice de precios de combustibles generado por MATLAB 92
Figura 5.14: Ajuste del índice de precios de insumos industriales generado por MATLAB 92
Figura 5.15: Ajuste del índice de precios de alimentos y bebidas generado por MATLAB 93
Figura 5.16: Ajuste del índice de precios de petróleo crudo generado por MATLAB 93
Figura 5.17: Ajuste del índice de precios de no combustibles generado por MATLAB 94
Figura 5.18: Ajuste del índice de precios general de mercancías generado por MATLAB 94

CAPÍTULO I

INTRODUCCIÓN

Contenido

Generalidades sobre el tema

Motivación

Descripción del problema

Objetivo general

Aportación de la tesis

Organización de la tesis

1.1. Generalidades sobre el tema

Todo problema del mundo real desde económico hasta científico y de los campos de la ingeniería, se enfrenta finalmente a una tarea común, a saber, optimización (Ahn et al., 2001; Ahn y Ramakrishna, 2002; Ahn, 2006).

El problema de descubrir la relación matemática entre variables medidas empíricamente y un fenómeno, es un problema importante en economía y otras áreas de la ciencia. En la práctica, los datos pueden ser ruidosos y puede no haber forma de expresar la relación implícita de la forma correcta. Algunos problemas de este tipo son algunas veces llamados problemas de identificación simbólica de sistemas, problemas de caja negra, problemas de minería de datos o problemas de modelado. Cuando el modelo descubierto se usa para predecir valores futuros del estado de las variables del sistema entonces es llamado problema de pronóstico.

Las relaciones matemáticas pueden ser expresadas por medio de fórmulas y ecuaciones matemáticas ordinarias; sin embargo, un programa de ordenador puede ofrecer una mayor flexibilidad al expresar una relación matemática arbitraria. Los programas de computadora empiezan con una o más entradas (las variables independientes del sistema), realizan varias operaciones con las variables (incluyendo operaciones aritméticas y operaciones de toma de decisiones condicionales) y producen ciertas salidas (las variables dependientes). En consecuencia, el problema de encontrar un modelo matemático para datos empíricos puede ser visto como el problema de encontrar un programa de ordenador que produce el valor observado en una variable dependiente como su salida, cuando obtiene valores de las variables independientes como entrada. Decimos que estos problemas son de regresión simbólica porque estamos buscando una expresión matemática, en su forma simbólica, que ajuste o aproximadamente ajuste ciertos datos obtenidos.

La regresión simbólica difiere de las convencionales (regresión lineal, regresión cuadrática, regresión exponencial, y otros tipos de regresión) en que la naturaleza del modelo es especificada de antemano por el usuario. En la regresión lineal convencional, por ejemplo, el usuario proporciona un conjunto de valores de varias variables

independientes y el correspondiente valor de la variable dependiente. El objetivo es descubrir un conjunto de coeficientes numéricos que junto con la variable o variables independientes minimicen alguna medida de error (tal como la raíz cuadrada de la suma de la diferencia de cuadrados) entre los valores obtenidos de la expresión lineal y los valores dados de la variable dependiente. De forma similar, en la regresión cuadrática el objetivo es descubrir un conjunto de coeficientes numéricos para una expresión cuadrática que minimice el error. Al emplear la técnica de regresión convencional, el usuario debe seleccionar, como un paso preparatorio, si se debe tratar de ajustar a un modelo lineal, a un modelo cuadrático o a algún otro modelo. Pero frecuentemente, el problema real es decidir qué tipo de modelo es apropiado para ajustar los datos y no sólo se basa en encontrar los valores precisos de los coeficientes numéricos después que el modelo ha sido seleccionado. La regresión simbólica busca ambos, la forma de la función y los coeficientes numéricos apropiados que van con la forma de la función. Encontrar la forma de la función y los coeficientes numéricos apropiados, puede ser equivalente a buscar en un espacio de posibles programas de ordenador, un programa particular que produzca el resultado deseado.

El programa de ordenador deseado puede ser encontrado por medio de la programación genética (PG), originalmente desarrollada para resolver problemas de inteligencia artificial, programación automática y aprendizaje automático. En la programación genética, las poblaciones de programas de ordenador (que desde ahora llamaremos soluciones) son evolucionadas usando competición darwiniana y operaciones genéticas. La competición darwiniana está basada en la supervivencia y la reproducción de los más adaptados. El operador de cruce o recombinación (reproducción sexual) son generalmente soluciones que se combinan en el proceso de generar nuevas generaciones (población con nuevas soluciones) que en el mejor de los casos son las mejor adaptadas. La solución única seleccionada después de muchas generaciones puede ser la solución satisfactoria al problema en cuestión.

1.2. Motivación

Los modelos matemáticos son nuestro medio para representar de forma discreta ciertos fenómenos que son difíciles de representar o de comprender en su forma real. El contar con un modelo nos permite estudiar y entender el fenómeno en una forma más simplificada.

Se cuentan con varias herramientas que nos permiten crear modelos matemáticos pero muchas de ellas utilizan técnicas poco versátiles que resultan insuficientes para realizar ajustes de curvas con la precisión requerida. La programación genética nos proporciona la flexibilidad que se necesita (a cambio de costo computacional) para crear funciones matemáticas que simulen el comportamiento de algún fenómeno a partir de ciertas variables de entrada. Esto nos permite construir modelos que no se podrían construir con las herramientas habituales.

1.3. Descripción de problema

Cuando se necesita crear un modelo matemático de algún fenómeno, se recurre a la técnica de ajuste de curvas que, mediante cierto proceso, crea una función matemática que simula el comportamiento del fenómeno en cuestión. Una forma habitual de hacerlo es mediante un ajuste polinomial; el método consiste en encontrar un polinomio de grado *n*, donde *n* es el número de puntos que se desean ajustar.

Esto hasta cierto punto es simple, pero al aumentar el número de puntos a ajustar, el problema se vuelve más complejo, convirtiéndose en un problema combinatorio donde se deben encontrar los valores precisos, tanto de los coeficientes como de las variables que formarán la función.

El objetivo es crear el modelo matemático mediante programación genética con los operadores necesarios para construir un polinomio. Este modelo estará compuesto de las variables necesarias para simular el comportamiento del fenómeno económico especificado. Las variables utilizadas son series de datos que representan otros fenómenos económicos, es decir, se construye un modelo que describe el

comportamiento de un fenómeno económico a partir de otros fenómenos que por defecto estarían correlacionados con éste.

1.4. Objetivos

1.4.1. Objetivo general

Encontrar un modelo matemático que describa el comportamiento de un fenómeno económico a partir de otras variables económicas que estén correlacionados con el fenómeno en cuestión.

1.4.2. Objetivos específicos

- Crear un sistema de programación genética en LISP dedicado al ajuste de curvas y aplicarlo a problemas con una sola variable.
- Mejorar el sistema de programación genética creado y aplicarlo a problemas de múltiples variables.
- Crear un método que analice las soluciones generadas por el sistema de programación genética y muestre la relevancia de cada variable incluida en la solución.
- Usar el sistema de programación genética y el método de análisis de soluciones con un conjunto de datos económicos.

1.5. Aportaciones de la tesis

Las aportaciones de la tesis son:

- Superamos los resultados de Kamal y Eassa (2002) con nuestro sistema de programación genética (GP Curve Fitting).
- El desarrollo de un programa para generar modelos económicos a partir de múltiples variables de entrada.

- La creación de un modelo matemático que permite un análisis de los datos a futuro.
- Un modelo matemático que sirve para técnicas de regresión simbólica, problemas de modelado y otros fines.
- No se utilizó ninguna biblioteca ni herramienta ya elaborada. El sistema se desarrolló totalmente desde un principio, por lo cual se puede adaptar al problema en el que se esté trabajando sólo modificando las funciones necesarias.

1.6. Organización de la tesis

La tesis está organizada en seis capítulos:

El primer capítulo contiene una introducción con la descripción general del problema de estudio, así como los objetivos: general y particulares, además de las aportaciones que se obtuvieron de este trabajo.

El segundo capítulo es un panorama del estado del arte donde se describen algunas aplicaciones relacionadas con el ajuste de curvas, así como algunos trabajos de otros investigadores que dedicaron tiempo y esfuerzo al estudio del ajuste de curvas.

En el capítulo tres se describe el marco teórico con la información necesaria para entender más a fondo la programación genética y el ajuste de curvas. Y todavía algo más importante; cómo aplicar la programación genética para resolver problemas de ajuste de curvas.

El cuarto capítulo es una breve explicación del desarrollo del proyecto. Muestra el uso de ciertas variables importantes y parámetros que se pueden ajustar. En general es una descripción del funcionamiento del sistema que se creó en esta investigación, el cual hemos llamado “GP Curve Fitting”.

El quinto capítulo es dedicado a la presentación de los resultados. En él mostramos algunos de los experimentos realizados en el transcurso de la investigación.

Por último en el sexto capítulo se exponen las conclusiones de la investigación y qué puntos proponemos para un estudio futuro.

CAPÍTULO II

ESTADO DEL ARTE

Contenido

En este capítulo presentaremos los trabajos realizados para el ajuste de curvas tanto con métodos deterministas como con métodos estocásticos. Veremos software comercial como MATLAB y software gratuito como LabFit que utilizan métodos deterministas para resolver ajustes. Por otra parte, también mostraremos los trabajos de otros investigadores que tratan de resolver problemas de este tipo con métodos alternos, como los algoritmos evolutivos. Explicaremos las características principales de cada trabajo, con el fin de armar un pequeño escenario de lo que se ha investigado sobre el tema del ajuste de curvas y las herramientas ya elaboradas.

2.1. Ajuste de curvas con métodos determinísticos

Existen diferentes herramientas para ajustar curvas con métodos determinísticos, tales como la regresión lineal, no lineal, logarítmica, polinomial etc. Estos métodos nos aseguran llegar a la mejor solución dentro de los límites del modelo que se esté utilizando. Sin embargo, al predefinir un modelo para el ajuste, también limitamos las soluciones posibles. Al limitarnos de esa manera, la mejor solución de ese modelo particular podría no ser una de las mejores soluciones en general.

Los métodos determinísticos son muy rápidos y precisos cuando el espacio de búsqueda es moderado; sin embargo, cuando el espacio de búsqueda es demasiado grande, se tornan lentos y en algunas ocasiones podrían tardar años en encontrar una solución óptima. Ejemplo de esto son los problemas NP-completos. El problema de la mochila (abreviado KP del inglés *knapsack problem*) es un problema NP-completo propuesto por Karp (1972); es considerado un problema de optimización combinatoria y si bien la formulación del problema es sencilla, su resolución es más compleja.

2.1.1. Herramienta de ajuste de curvas de MATLAB

La herramienta de ajuste de curvas de MATLAB proporciona las técnicas más utilizadas para ajustar las curvas y las superficies de los datos, incluidos los de regresión lineal y no lineal. La herramienta soporta opciones de regresión robusta en la que se encuentran valores atípicos.

La aplicación de ajuste de curvas simplifica tareas comunes como:

- Importar datos desde el espacio de trabajo.
- Visualizar datos para realizar el análisis exploratorio de los datos.
- Generar ajustes utilizando múltiples algoritmos.
- Evaluar la precisión de los modelos.

Trabajar con la línea de comandos permite desarrollar funciones personalizadas para el análisis y la visualización. Estas funciones permiten:

- Duplicar el análisis con un nuevo conjunto de datos.
- Replicar el análisis como múltiples conjuntos de datos.
- Insertar una rutina de ajuste en las nuevas funciones MATLAB.

La herramienta de ajuste de curvas de MATLAB provee una sintaxis simple para el ajuste por línea de comandos, como se muestra en los siguientes ejemplos:

- Regresión lineal: `fittedmodel = fit ([X,Y] , Z , 'poly11');`
- Regresión no lineal: `fittedmodel = (X , Y , 'fourier2');`
- Interpolación: `fittedmodel = fit([Time,Temperature], Energy, 'cubicinterp');`
- Suavizado: `fittedmodel = fit([Time,Temperature], Energy, 'lowess', 'span', 0.12);`

Los resultados de la operación de ajuste son almacenados en un objeto llamado "`fittedmodel`", los análisis como graficado, evaluación, y cálculo de derivadas e integrales puede ser realizado aplicando el método a este objeto, por ejemplo:

- Graficado: `plot (fittedmodel)`

La herramienta de ajuste de curvas soporta regresión lineal y no lineal incluyendo los siguientes modelos:

- Líneas y planos
- Polinomios de alto grado
- Series de Fourier
- Funciones Gaussianas
- Funciones de Weibull
- Funciones exponenciales
- Funciones racionales
- Suma de senos

2.1.2. Ajuste de curvas con Microsoft EXCEL

El programa Excel de Microsoft es una hoja de cálculo que tiene una herramienta para el ajuste de curvas. Para realizar un ajuste tenemos que seleccionar alguno de los siguientes modelos: exponencial, lineal, logarítmico, potencial y polinomial. En caso de que se seleccione el ajuste polinomial, se permite usar polinomios desde grado dos hasta grado seis. Una vez seleccionado el modelo, se buscan automáticamente los valores de los coeficientes con los que se pueda optimizar el ajuste.

Excel es una buena elección cuando se desea realizar el ajuste de datos a partir de un conjunto de datos que no tenga demasiadas inflexiones; de otra forma el ajuste no resulta muy bueno.

Como en todo método determinístico, se tiene la desventaja de que el modelo permanece igual y sólo se cuenta con la incertidumbre de encontrar los coeficientes adecuados para lograr el mejor resultado posible.

2.1.3. Ajuste de curvas con LabFit

El software LabFit fue desarrollado con el objetivo del tratamiento y análisis de datos experimentales.

Este software cuenta con un menú para ajuste de curvas y los principales programas de este menú usan regresión no lineal. LabFit ajusta funciones de una o más variables independientes (de una hasta seis). Las bibliotecas de este software cuentan con más de 200 funciones con una variable independiente y con casi 280 funciones con 2 variables independientes. El usuario dispone de un programa buscador de funciones. Si es necesario existe una opción para que el usuario sea capaz de escribir su propia función de ajuste. Una vez determinados los parámetros de ajuste, es posible extrapolar la función de ajuste y, para dos y tres dimensiones, se muestra la gráfica de la función.

Podemos concluir que LabFit cuenta con un número de funciones predeterminadas con cierto número de variables independientes. Esto le da al usuario un gran número de

alternativas a considerar, por lo que él debe conocer la forma correcta de solucionar el problema o intentar solucionarlo con cada una de las alternativas.

2.2. Ajuste de curvas con heurísticas

Las heurísticas son una forma de abordar problemas cuando la solución se encuentra en un espacio de búsqueda demasiado grande, donde un método habitual podría tardar demasiado tiempo en llegar a la solución óptima. Las heurísticas no aseguran encontrar la mejor solución; en cambio, nos proporcionan una salida cercana a la mejor. No obstante, en algunas ocasiones es poco probable, obtener resultados demasiado erróneos. A pesar de ello, las heurísticas han resuelto muchos problemas de manera exitosa donde los métodos convencionales fallaron.

Existen muchos algoritmos heurísticos como la programación genética, algoritmos genéticos, escalado de colinas, evolución diferencial, colonias de hormigas, etc., algunos de ellos inspirados en la naturaleza.

Los algoritmos genéticos (AG) y la programación genética (PG) son algoritmos bio-inspirados basados en la teoría de la selección natural, donde los individuos más aptos son los que se reproducen y proliferan, mientras que los débiles se extinguen.

2.2.1. Aplicación de algoritmos genéticos en ajuste de curvas

Este trabajo fue realizado por Senturk (2009). Fue realizado para obtener el grado de maestro en ciencias de la computación.

El objetivo de la tesis, como en la mayoría de trabajos en donde se usan algoritmos genéticos, es encontrar el valor de los coeficientes de un polinomio que minimice el error sobre los datos para ajustar cierta curva.

Presentamos un experimento de su tesis donde ajusta una serie de puntos generados por la función $y = x^2 - 2x + 1$. La misión del algoritmo genético es redescubrir los

coeficientes de la función original o encontrar otros que generen los mismos puntos o puntos parecidos a la función original.

El programa se corrió con una población de 100 individuos, probabilidad de recombinación de 0.5, probabilidad de mutación de 0.1, precisión de 4 decimales, y un máximo de 1000 generaciones.

El resultado del ajuste es mostrado en la Figura 2.1 tomada de la tesis de Senturk (2009), donde podemos observar la representación gráfica de la función que se ajustó y también el ajuste en sí.

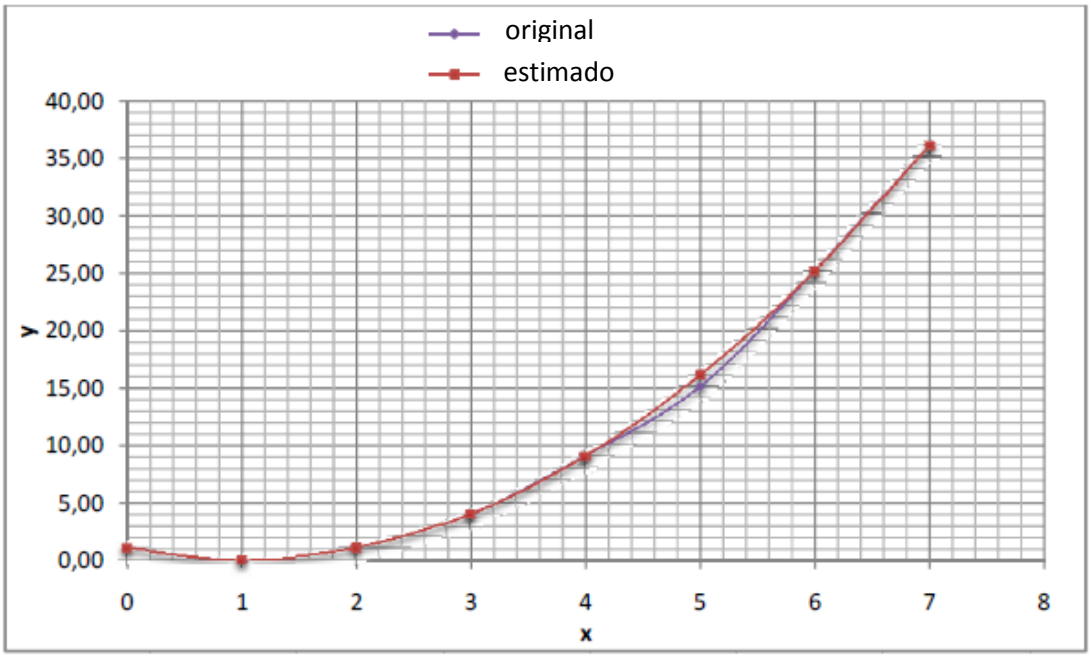

Figura 2.1: Curva estimada y curva original.

El error que se obtuvo fue de 0.009 y los coeficientes fueron: C1 = 1.0037, C2 = -2.0030, C3 = 1.0046. Entonces la ecuación queda de la siguiente manera $y = 1.0037x^2 - 2.0030x + 1.0046$.

Cuando los datos de la primera curva a ajustar son aplicados a un polinomio de grado tres (recordemos que en un algoritmo genético se debe especificar la función con la que se va a trabajar), el resultado obtenido tiene un error de 0.232. El error aumenta puesto que el espacio de búsqueda también aumentó, ya que ahora se busca el valor de cuatro coeficientes. Los valores de los coeficientes quedan de la siguiente manera: C1 = 0.0083, C2 = 0.8732, C3 = -1.4137, C4 = 0.2527. Esta vez los resultados no fueron aceptables, por lo cual el programa vuelve e regenerar la población y restablece las generaciones con el fin de obtener mejores resultados. El error es reducido a 0.1625 y los coeficientes ahora son C1 = 0.0010, C2 = 0.9732, C3 = -1.7628, C4 = 0.6449.

Notemos que a pesar de que sólo aumenta un coeficiente a buscar, el error aumenta considerablemente. Esto es porque ahora hay que encontrar cuatro números en el espacio de los números reales (en un rango limitado) que sustituidos en el polinomio nos generen un ajuste aceptable. Esto implica que entre más coeficientes busquemos el problema se vuelve más complejo.

El éxito de los métodos de optimización distintos a los algoritmos genéticos depende del punto inicial de la estimación. Por ejemplo, con el fin de determinar el punto inicial cuando se usa el método de Gauss-Newton para la búsqueda, se requiere un estudio preliminar.

Como se presentó en este estudio, los algoritmos genéticos no requieren ninguna información auxiliar o un estudio preliminar de los datos. El programa sólo necesita saber el número de generaciones, el tamaño de la población, establecer una probabilidad de mutación y una probabilidad de recombinación. Una vez establecidos esos parámetros, el investigador puede aceptar una solución en un determinado tiempo y detener el programa.

2.2.2. Algoritmos genéticos, un enfoque para el ajuste de curvas

Gulsen et al. (1995) escribieron un trabajo llamado “A genetic algorithm approach to curve fitting”. Como el título lo indica se aborda un problema de ajuste de curvas con una heurística.

Se presenta un algoritmo genético que ajusta curvas por medio de polinomios. El objetivo del ajuste de curvas es seleccionar los coeficientes adecuados con los cuales se minimice el error total sobre el conjunto de datos que están siendo considerados. Una vez que la forma de la función y una medida de error han sido seleccionadas, el ajuste de curvas se convierte en un problema de optimización sobre el conjunto de datos establecidos. Desde que los algoritmos genéticos han sido usados exitosamente como técnicas de optimización global para funciones continuas y problemas combinatorios, ellos parecen adecuados para el ajuste de curvas cuando se estructura como un problema de selección de parámetros.

El proceso comienza codificando un valor real para el algoritmo genético, así que cada solución es codificada en un vector de coeficientes de valores reales. En un principio experimentaron brevemente con la tradicional codificación binaria, donde todos los coeficientes son representados en base 2; sin embargo, esto probó ser muy deficiente y no se obtuvieron buenos resultados. La evolución del algoritmo usa dos operaciones básicas: recombinación y mutación. La recombinación genera nuevos coeficientes combinando la información genética (soluciones en código binario) de dos soluciones 'padre'. La mutación modifica cierta solución aplicando un pequeño cambio en la información genética.

2.2.2.1. Generación de la población inicial

Cada miembro de la población es un vector de coeficientes reales. En todos los experimentos se usó una población de 50 soluciones. Tomando como ejemplo la siguiente ecuación:

$$y = C \quad + C_2 x_1^3 + C_3 cos X_2 + C_4 x_1 x_2 \qquad (1)$$

Cada solución tiene cuatro números reales, C1, C2, C3, C4. Los valores iniciales de cada coeficiente son generados aleatoriamente de una distribución uniforme sobre rangos especificados para cada coeficiente

2.2.2.2. Mecanismo de evolución

La función objetivo, la cual mide qué tan bien la curva propuesta se ajusta a los datos reales, es calculada para cada miembro de la población. Se consideran tres diferentes medidas en su estudio: error cuadrático, error absoluto y error máximo. El valor calculado es usado para la selección de soluciones para el engendramiento de la siguiente generación.

No se usó el método común de selección por ruleta (Goldberg, 1989) por las grandes variaciones en los valores de aptitud entre los miembros de la población. Un método basado en la selección por ruleta tiende a fomentar la convergencia prematura debido a que se enfoca en la vecindad del mejor individuo encontrado, haciendo que la población se vuelva muy similar. Se seleccionan los padres para crear la descendencia uniforme de los mejores 25 miembros de la población actual, esto es, de la mitad superior de la población. Esta es una forma estocástica de selección por truncamiento originalmente usada de forma determinista en estrategias de evolución para la evolución elitista (Back y Schwefel 1993). Muhlenbein y Schlierkamp-Voosen (1993) después usaron la versión del método estocástico en su algoritmo genético. Los valores de los coeficientes descendientes son determinados calculando la media aritmética de los correspondientes coeficientes de los dos padres. Este método de cruce es conocido como recombinación intermedia ampliada con $\alpha = 0.5$ (Muhlenbein y Schlierkamp-Voosen, 1993). Esto mantiene los valores de los coeficientes de los padres idénticos en los hijos, mientras que promedia los coeficientes no idénticos. Se muestra un ejemplo que hace referencia a la ecuación 1 en la Tabla 2.1.

Tabla 2.1: Ejemplo de recombinación intermedia ampliada con $\alpha = 0.5$

Padre A	45.876	32.958	12.098	-3.892
Padre B	12.988	35.832	0.234	-12.984
Descendencia	29.432	34.395	6.166	-8.438

La mutación altera las soluciones y con esto pueden ser exploradas nuevas regiones del espacio de búsqueda. En el proceso de mutación, primero se calcula el rango de cada coeficiente determinando el valor máximo y mínimo de cada uno en particular en la población actual. Entonces, el valor alterado se obtiene multiplicando el rango por un factor. Este factor es una variable aleatoria uniformemente distribuida entre $\pm k$ (rango del coeficiente). Se usa el valor de k=2 para las pruebas. El esquema de mutación está relacionado con el que usaron Muhlenbein y Schlierkamp-Voosen (1993) en su algoritmo genético. Sin embargo, el rango que usaron fue estático, mientras que el que usaron Gulsen et al. (1995) se adapta de acuerdo a los valores extremos encontrados en la población actual. Se mutaron todos los coeficientes de 25 miembros de una población de 50 seleccionados con probabilidad uniforme.

Después de la mutación y el cruce, la descendencia modificada es agregada a la población para formar un total de 100 soluciones (50 de la población anterior, 25 del proceso de cruce y 25 del proceso de mutación). Se evalúa la aptitud de las 100 soluciones y las peores 50 son sacrificadas para iniciar el nuevo ciclo.

2.2.2.3. Resultados obtenidos

Para evaluar el algoritmo genético se realizaron tres diferentes problemas de prueba de los cuales, dos son tomados de estudios previos de algoritmos genéticos para ajuste de curvas.

Problema de prueba 1: El primer problema involucra la estimación de dos coeficientes (C1 y C2) de una recta dada por la ecuación 2. Este problema fue estudiado por Karr et al. (1991). Es un problema muy sencillo para ser usado como una introducción a problemas más complejos.

$$y = C_1 x + C_2 \qquad (2)$$

En este problema, la forma de la función, los coeficientes, los límites de búsqueda iniciales y el conjunto de datos fueron tomados directamente del trabajo de Karr et al. (1991). En este problema de prueba fueron usados seis puntos de datos para ajustar y los

valores verdaderos de C_1 y C_2 son 1 y 0 respectivamente. Los límites iniciales superior e inferior de los coeficientes fueron puestos en ± 32. El algoritmo fue ejecutado en cuatro ocasiones con diferentes números-semilla aleatorios. Se usó la métrica de error cuadrático como función de aptitud.

Tabla 2.2: Coeficientes encontrados por el algoritmo genético y coeficientes encontrados en el experimento de Karr et al. (1991)

Prueba	C1	C2	Error cuadrático
1	0.994865	0.054508	0.010059
2	1.006897	-0.015690	0.003333
3	1.012757	-0.066320	0.013703
4	1.004253	-0.022980	0.001614
Media de los 4	1.004698	-0.012621	0.007177
Karr et al., 1991	0.968800	0.000000	0.127000

Los números finales de C_1 y C_2 y el error cuadrático después de 6 generaciones son mostrados en la Tabla 2.2 donde se incluyen los coeficientes y el error cuadrático encontrado después de 6 generaciones con el algoritmo genético. Además, ambos, el ajuste de Karr et al. (1991) y el ajuste elaborado por Gulsen et al. (1995) dieron buenos resultados en cuanto al valor de los coeficientes encontrados después de pocas generaciones. Los resultados de Gulsen et al. (1995), para este problema en particular, resultaron mejores que los de Karr et al. (1991).

Problema de prueba 2: El segundo problema mostrado en la ecuación 3, tiene 10 variables independientes y 9 coeficientes que deben ser estimados. Este problema fue estudiado por Rogers (1991) y antes por Friedman (1988). Fueron usados con un conjunto de 200 datos seleccionados aleatoriamente; el problema fue diseñado de tal forma que la respuesta dependiera sólo de 5 variables. Las variables restantes de x_6 a x_{10}, no tienen efecto en la respuesta de la función, y son incluidas en el conjunto de datos como ruido. Por lo tanto, los valores óptimos de C_5 hasta C_9 deberían ser cero.

$$y = C_1 \sin(\pi x_1 x_2) + C_2(x_3 - 0.5)^2 + C_3 x_4 + C_4 x_5 + \sum_{n=6}^{10} C_{n-1} X_n \qquad (3)$$

Los coeficientes finales encontrados por el AG con respecto a los valores correctos son mostrados en la Tabla 2.3. En este problema fueron usados 100 datos para ajustar, y los límites superior e inferior de la búsqueda inicial fueron ± 10. Se necesitaron 250 generaciones para lograr resultados aceptables.

Tabla 2.3: Coeficientes encontrados después de 250 generaciones. Comparación de valores verdaderos y valores encontrados por el algoritmo genético (AG).

Coeficiente	Valor verdadero	Valor del AG
1	10	10.000023
2	20	19.999958
3	10	10.000037
4	5	4.999990
5	0	0.000012
6	0	0.000000
7	0	-0.000003
8	0	0.000013
9	0	0.000019

Problema de prueba 3: El tercer problema de prueba es una ecuación de segundo grado de tres variables. La función, que se muestra en la ecuación 4, tiene 10 coeficientes que tienen que ser estimados y afirman que es más grande y compleja que todas las que se habían estudiado hasta ese momento en problemas de ajuste de curvas con algoritmos genéticos. Se emplea un mecanismo de reproducción secuencial, la cual divide los coeficientes en dos grupos. El primer grupo (está formado por los primeros 4 coeficientes de la ecuación 5): el segundo grupo por los coeficientes restantes.

$$y = C_1 + C_2 x_1 + C_3 x_2 + C_4 x_3 + C_5 x_1^2 + C_6 x_2^2 + C_7 x_3^2 + C_8 x_1 x_2 + C_9 x_1 x_3 + C_{10} x_2 x_3 \qquad (4)$$

En este problema se intentan ajustar 25 datos. El rango para los coeficientes es de ± 500 y se necesitaron 2500 generaciones para obtener resultados aceptables. Los resultados se muestran en la Tabla 2.4.

Tabla 2.4: Coeficientes encontrados después de 2500 generaciones por el algoritmo genético

Coeficiente	Ejecución 1	Ejecución 2	Ejecución 3
1	9.986	9.998	10.002
2	9.999	10.000	10.000
3	10.000	10.000	10.000
4	10.000	10.000	10.000
5	10.000	10.000	10.000
6	10.000	10.000	10.000
7	10.000	10.000	10.000
8	10.000	10.000	10.000
9	10.000	10.000	10.000
10	10.000	10.000	10.000
Error cuadrático	0.0017	0.000	0.000

2.2.3. Resolución de ajuste de curvas con programación genética

La programación genética (PG) es una rama de los algoritmos genéticos. La principal diferencia entre ellos es la forma en que representan la solución. En los algoritmos genéticos la representación común es en código binario, mientras que en la programación genética se usan árboles. Los árboles pueden almacenar expresiones matemáticas, conteniendo terminales y operadores en sus nodos.

Tomaremos como referencia el artículo elaborado por Kamal y Eassa (2002) llamado "Solving curve fitting problems using Genetic Programming", en el cual se realizan ajuste de curvas con conjuntos pequeños de datos experimentales usando programación genética.

El sistema de programación genética de Kamal y Eassa (2002) se desarrolló en Visual C++ con metodología orientada a objetos en un sistema operativo Windows. La interfaz fue diseñada separada del núcleo de PG para que este último sea flexible y se pueda usar en cualquier otra aplicación.

Kamal y Eassa recalcan que al usar programación genética para el ajuste de curvas en problemas de regresión lineal, no se necesita saber la forma o grado de la función con la que vamos a tratar de igualar los valores de la variable dependiente, debido a que en PG se pueden generar diversas expresiones y encontrar la mejor que pueda obtener el sistema después de cierto número de iteraciones.

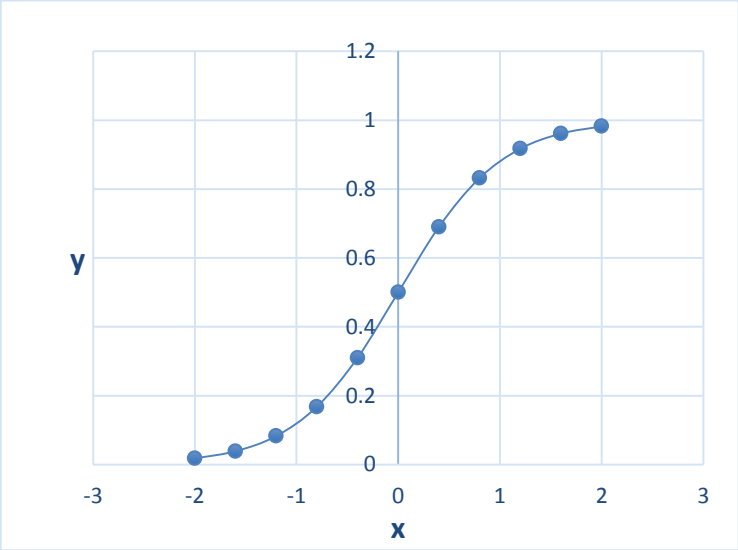

Figura 2.2: Representación gráfica de datos experimentales para el ajuste de curvas

Mostramos a continuación un experimento realizado por los autores mencionados, en donde se ajusta el conjunto de datos mostrado en la Tabla 2.5. El conjunto de datos se puede ver gráficamente en la Figura 2.2. Los resultados son medidos con la función de aptitud f = 1 / (1+S) donde S es la fórmula de mínimos cuadrados. Después de cierto número de iteraciones (no mencionadas en el artículo), se obtiene una solución con aptitud de 0.999. Por la aptitud de la solución se puede concluir que el experimento fue exitoso y que la PG puede resolver problemas de regresión lineal.

Tabla 2.5: Datos experimentales para el ajuste de curvas

x	-2.0	-1.6	-1.2	-0.8	-0.4	0.0	0.4	0.8	1.2	1.6	2.0
y	0.018	0.039	0.083	0.168	0.310	0.500	0.690	0.832	0.917	0.961	0.982

Tomando en cuenta los datos que proponen, se puede observar que el experimento es bastante básico, por lo que se puede inferir que el artículo no intenta decir que la heurística pueda superar o remplazar a los métodos determinísticos. Kamal y Eassa nos brindan una alternativa para realizar un ajuste de curvas.

2.2.4. Ajuste de curvas usando algoritmos genéticos

Messa y Lybanon (1991) resuelven ajuste de curvas mediante algoritmos genéticos (AG) en el artículo llamado "Curve Fitting Using Genetic Algorithms". Este artículo describe las investigaciones realizadas en el laboratorio de investigación oceanográfica y atmosférica en la aplicación de algoritmos genéticos para encontrar modelos matemáticos que describan ciertos datos experimentales.

Los algoritmos genéticos son técnicas de búsqueda basadas en mecanismos de selección natural. Estos han sido aplicados exitosamente en muchas aplicaciones debido a que son robustos y por su habilidad de buscar soluciones a problemas con grandes espacios de búsqueda. En particular los algoritmos genéticos son usados en ajuste de curvas. El AG selecciona los coeficientes de una función particular de tal forma que los puntos generados por la función pasen lo más cerca posible de los datos experimentales.

Las soluciones candidatas son vectores de números reales que representan los coeficientes de la curva que será modelada. En consecuencia cada solución candidata corresponde a una nueva función. Como tal, cada candidato a solución es evaluado con la fórmula de mínimos cuadrados. La evaluación de cada candidato con respecto a su aptitud guía al AG hacia la solución con mayor mérito.

Varios ejemplos de la aplicación de algoritmos genéticos a problemas de ajuste de curvas son presentados por Messa y Lybanon (1991). La convergencia a la solución

óptima es rápida cuando los coeficientes están en el rango establecido en el AG. Cuando se conoce poco acerca de los coeficientes del problema, entonces es necesario cierto grado de experimentación para resolverlo.

2.2.4.1. La representación

Los organismos en los algoritmos genéticos representan las soluciones al problema. En este caso las soluciones son valores reales asignados a cada coeficiente en el modelo de curva. Hay también una medida de la bondad del ajuste con respecto a los datos experimentales D. En consecuencia, si $f = f(a_1, a_2, \dots, a_n : x)$ es la curva que va a ser ajustada, entonces, el valor real para los coeficientes $a_1, a_2, \dots, a_n$ es el que va a ser buscado. Por ello, los vectores $r_1, r_2, \dots, r_n$, donde a_1 es reemplazada por números reales r_i, $i = 1, 2, \dots, n$, son soluciones candidatas. Por lo tanto, los organismos para el AG usados en el ajuste de curvas son vectores de números reales $r_1, r_2, \dots, r_n$.

Este modo de representación es muy útil desde el punto de vista de alto nivel de resolución del ajuste de curvas. Sin embargo, el AG trabaja a bajo nivel, al nivel de bits o alelos. Para tener éxito, el AG considera una representación de números reales r_i en forma de alelos. Dado un límite superior e inferior para cada r_i, u_i y l_i respectivamente, podemos ver al vector r_i como un entero binario sin signo con *m* bits al calcular su valor con respecto a u_i y l_i.

Dado un entero binario b, donde b está en el rango $[0, 2^{k-1}]$, su correspondiente valor binario es dado por la fórmula:

$$r = \frac{b}{2^m} * (u - l) + l, \qquad (5)$$

donde $u\ y\ l$ son el límite superior e inferior respectivamente. Combinando estos dos niveles una solución candidata se construye:

$$\beta = \langle b_{11}\, b_{12}\, b_{13} \dots\ b_{1m}, b_{21}\, b_{22}\, b_{23} \dots\ b_{2m}, b_{n1}\, b_{n2}\, b_{n3} \dots\ b_{nm} \rangle,$$

donde cada entero binario $, b_{i1}\, b_{i2}\, b_{i3} \dots\ b_{im}$ corresponde a un número real r_i, que se encuentra en el intervalo $[\, l_i, u_i\,]$. La correspondencia es dada en la ecuación 6.

Calcular el valor de aptitud de una solución candidata β requiere de dos pasos: convertir cada entero binario $b_{i1}\, b_{i2}\, b_{i3} \ldots\, b_{im}$ en su correspondiente valor real r_i y evaluar la curva $f = f(a_1, a_2, \ldots, a_n; x)$ con respecto a los datos experimentales D.

2.2.4.2. La función de aptitud

La función de aptitud es una forma de medir qué tan bien ajusta la solución candidata a los datos experimentales. Esta es la entidad que guía al AG hacia la solución. La aptitud usada en este artículo es construida en varios pasos:

- La solución candidata β es convertida en un vector de números reales $r = \langle r_1, r_2, \ldots, r_n \rangle$ y el valor de $f(r; x) = f(r_1, r_2, \ldots, r_n; x)$ es calculado por cada x_i donde la pareja (x_i, y_i) está en D.
- La suma de la diferencia de cuadrados entre $f(r; x_i)$ y y_i genera una medida donde 0 indica un ajuste exacto. Este valor es llamado pre aptitud de β. Así,

$$preaptitud(\beta) = \sum (f(r; x_i) - y_i)^2, \quad (6)$$

  donde en la sumatoria intervienen todos los valores (x_i, y_i) de D.

- Como los algoritmos genéticos trabajan buscando los máximos valores y la función de preaptitud maneja los valores menores como un mejor ajuste, el máximo y el mínimo deben ser intercambiados. Para toda la población, max_p representa el valor más grande de aptitud (esto es, el peor ajuste). Se calculó la aptitud-en-bruto como la diferencia:

$$AptitudEnBruto(\beta) = max_p - preaptitud(\beta), \quad (7)$$

  Este cálculo no es precisamente la medida de aptitud usada en el AG. Muchos AG requieren un escalamiento de los valores de aptitud. El escalamiento amplifica la distinción entre organismos buenos y muy buenos. En los casos más simples el escalamiento lineal es usado.

$$Aptitud(\beta) = m_p * AptituEnBruto(\beta) + b_p, \quad (8)$$

donde m_p y b_p son constantes calculadas para cada población p.

2.2.4.3. Selección

El método de selección que fue usado fue el muestreo estocástico sin reemplazamiento, llamado "valor esperado" por Goldberg (1989). Para implementar este método, dado un organismo β con aptitud f_i, la aptitud es ponderada con respecto al promedio de la población f^*. La división truncada de f_i y f^* da el número esperado de la descendencia formada por β. El número esperado de descendencia de cada organismo en la población es calculado. Si algún lugar en la nueva población queda vacío después que el proceso es completado, la parte fraccionaria de cada aptitud, truncada por la división previa, es usada para determinar las posiciones restantes.

Como mejora a este método de selección, la estrategia de elitismo de De Jong (1975) fue usada, por lo que el mejor organismo de la generación es copiado a la siguiente sin hacerle ningún cambio. Esta estrategia le da un poco más de peso al mejor organismo de lo que pudo haber obtenido con el proceso de selección puro y previene que el mejor organismo pueda ser perdido con el proceso de cruce y mutación.

2.2.4.4. Recombinación

Un método de cruce simple fue usado y al parecer trabajó bien. Para algunos problemas con demasiados coeficientes o donde muchos bits son requeridos para una mejor precisión, el cruce en dos puntos dio mejor resultado que el cruce en un solo punto. Se le aplicó la operación de cruce al noventa por ciento de las parejas seleccionadas y el diez por ciento restante fue copiado a la siguiente generación sin sufrir cambios.

2.2.4.5. Mutación

El método de mutación más comúnmente usado consiste en cambiar el valor de un bit en una posición del gen con una frecuencia igual a la probabilidad de mutación. Se usaron

muchas probabilidades de mutación en los experimentos realizados, pero se obtuvieron pocos resultados buenos con el método de cambio de bit. Tasas altas de mutación tuvieron el efecto de introducir mucho material nuevo dentro de la población. Así, buenas soluciones eran encontradas frecuentemente. Sin embargo, una tasa de mutación alta también tiene el efecto de cambiar frecuentemente a los individuos incluyendo a los que tenían buena aptitud. Así que, mientras tasas altas de mutación encontraban buenas soluciones, éstas tendían a distorsionar a toda la población. Hubo algunas veces en las que aunque se estableciera una tasa alta de mutación el AG no obtenía una solución aceptable. Con una tasa baja de mutación el AG frecuentemente fallaba.

Un problema con este método de mutación está relacionado con lo llamado "escalado de colina". Supongamos que un coeficiente óptimo es el valor 0.5 y es representado como 100000 usando enteros binarios de seis bits y un intervalo de [0, 1]. Una solución candidata de 011111 (=0.46875) podría dar resultados bastante buenos debido a su cercanía con el óptimo. Sin embargo, una vez que el AG ha empezado a converger a la solución subóptima, es muy poco probable que se mueva de la solución subóptima a la óptima.

A diferencia del método de mutación por cambio de bit, un método que agrega un valor real α a los organismos (o sustrae) puede resultar más eficiente. El valor de α es una potencia de 2 de 1 hasta 2^m. Si $\alpha_i = 0..010..0$ con un uno en la i-ésima posición y cero en las demás posiciones, entonces, éste método de mutación agrega o sustrae α del organismo que va a ser mutado, donde α es $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$.

El método procede de la siguiente manera: si aleatoriamente se ha determinado que el alelo en la posición relativa i (el *i*-ésimo gen de algún coeficiente en β) va a ser mutado, entonces, α_i es o bien agregado o sustraído de β. Si el i-ésimo gen es 0, entonces agregar α_i es idéntico al cambio por bit. O si el i-ésimo gen es 1, entonces la sustracción es idéntica a la mutación de cambio de bit. La mejora de este método se observa cuando α es agregado en un gen con valor 1 (o sustraído de un gen con valor 0). Dado el ejemplo de referencia, agregar un 1 al sexto gen en 011111 arroja el valor óptimo de 100000.

2.2.4.6. Resultados

La población se mantuvo en 100 individuos para todos los experimentos excepto para aquellos en los que la función era muy compleja. Para esos casos la convergencia era muy lenta y el rendimiento mejoraba considerablemente con 200 individuos.

La tasa de mutación que se adoptó fue de 1% para los experimentos que se mostrarán más adelante. El escalamiento fue crucial, ya que muchos valores de preaptitud eran muy cercanos entre sí. El valor de aptitud de un organismo es el nuevo valor después de aplicar el escalamiento a la aptitud en bruto.

El primer experimento que se muestra en el artículo que se describe ahora, es un ajuste de una función lineal $f = Ax + B$, con un excelente resultado dado que el problema es demasiado trivial y lo usan sólo como introducción y para una mejor comprensión de los experimentos siguientes. Los datos a ajustar son $D = \{(1.0, 0.0),\ (2.01, 2.15),\ (3.08, 4.14),\ (4.22, 6.08),\ (5.0, 8.17)\}$.

*Tabla 2.6: Coeficientes encontrados por el algoritmo genético en 50 generaciones. (evaluado en 100 generaciones)**

Intervalo	A	B	Error
Corto	197.849	-195.012	0.411948
Largo	197860	-195.047	0.411945
Largo*	197.853	-195.028	0.411944

El AG usado en el ajuste de curvas requiere que se establezcan ciertos límites para la búsqueda de coeficientes. Si el tamaño del intervalo es pequeño entonces la convergencia es rápida. Para intervalos grandes la convergencia se torna muy lenta. En este problema ya se conocían los valores de los coeficientes por adelantado $A = 2.0$ y $B = -2.0$, por esa razón se eligieron los intervalos [1, 3] y [-3] para A y B respectivamente. Se obtuvo una convergencia muy rápida encontrando los valores óptimos. En la Tabla 2.6 se pueden ver algunos resultados del algoritmo genético con intervalos pequeños e intervalos grandes.

Tabla 2.7: Coeficientes encontrados por el algoritmo genético en 300 generaciones

Coeficiente	Ejecución 1	Ejecución 2	Ejecución 3
A	0.500001	0.499999	0.499999
B	0.249999	0.250000	0.250000
C	0.249994	0.250008	0.250004
D	0.750004	0.750000	0.749996
Error	1.73 E-5	1.49 E-5	2.30 E-5

Ahora veremos un experimento un poco más complejo con un polinomio de grado 3 $f = Ax^3 + Bx^2 + Cx + D$, el cual tiene que ajustar siete puntos. Los coeficientes aproximados fueron encontrados en tres ejecuciones de 300 generaciones. El experimento llegó a una solución razonablemente buena. En la Tabla 2.7 mostramos los resultados más detallados de este experimento.

Tabla 2.8: Coeficientes encontrados por el algoritmo genético en 500 generaciones

Coeficiente	Ejecución 1	Ejecución 2	Ejecución 3
A	0.257026	0.257026	0.257033
B	-0.047801	-0.047803	-0.047792
C	-1.504.910	-1.504.920	-1.505.000
D	1.048.910	1.048.910	1.048.330
E	0.185663	0.185687	0.185813
Error	0.967972	0.967972	0.967972

Veamos ahora cómo se comporta el algoritmo genético con el polinomio de grado 4 $f = Ax^4 + Bx^3 + Cx^2 + Dx + E$. Como es de esperarse, la precisión del AG disminuye si el intervalo de búsqueda de coeficientes es demasiado grande. Ahora se necesitaron 500 generaciones para lograr valores aceptables. El AG fue probado con intervalos grandes para encontrar los valores aproximados y después se acortó el intervalo para obtener una mejor precisión. Por ejemplo, para empezar se fijó un intervalo [-10, 10]

para el coeficiente A, y después de muchos experimentos se volvió evidente que el intervalo debería ser [0.20, 0.30]. En consecuencia, ejecutando el AG con los nuevos intervalos se obtuvo una mejor precisión.

Los resultados de este experimento se muestran en la Tabla 2.8. Podemos ver que el error aumentó considerablemente, debido a que el espacio de búsqueda se eleva considerablemente cada vez que agregamos un nuevo coeficiente a buscar.

2.3. Conclusiones

El uso de heurísticas y de métodos deterministas para realizar ajuste de curvas está bien documentado. Existen herramientas deterministas (MATLAB, LabFit) que realizan ajustes con distintos modelos matemáticos. También han sido implementados algoritmos genéticos que dado un modelo matemático (generalmente un polinomio) buscan coeficientes para mejorar el ajuste. Por otra parte, existen sistemas de programación genética que no necesitan que se especifique de antemano un modelo matemático para realizar el ajuste de curvas.

El problema que surge al usar los métodos ya existentes es que sólo involucran una variable independiente en la solución, por ello la solución encontrada no brinda información acerca de otras variables que pueden estar involucradas. Por ejemplo, contamos con 3 variables (que representan series de tiempo), el precio del gas natural, el precio del petróleo y el precio de la gasolina. Supongamos que necesitamos crear una solución que pueda involucrar las tres variables. Esto no es posible con las herramientas implementadas hasta el momento. Necesitamos una herramienta que logre involucrar *n* variables independientes.

MATLAB, LabFit, entre otras herramientas deterministas, son poco flexibles y no permiten lograr nuestro propósito, además del hecho que son de código cerrado. Por lo tanto, optamos por enfocarnos en las heurísticas para resolver nuestro problema. La mayoría de heurísticas necesitan que se elija a priori un modelo matemático lo cual dificulta la tarea de crear una función matemática que incluya distintas variables

independientes. En cambio la programación genética, crea funciones matemáticas a partir de operadores básicos y terminales, es decir, no necesita que se proponga un modelo matemático para realizar un ajuste de curvas. Por ello, creemos que la solución es mejorar y adaptar un sistema de programación genética que sirva para el problema en cuestión.

CAPÍTULO III

PROGRAMACIÓN GENÉTICA Y AJUSTE DE CURVAS

Contenido

Este capítulo contiene los conocimientos básicos para poder entender el funcionamiento de un sistema de programación genética (particularmente usado para el ajuste de curvas). Para alguien que se está iniciando en el área de los algoritmos evolutivos y sea de su agrado la programación genética, aquí podrá encontrar información ligera para después, una vez entendidos los principios básicos, poder asimilar información más compleja.

3.1. Introducción a la programación genética

El término programación genética (PG) tiene dos posibles significados. En primer lugar, se utiliza para englobar a todos los algoritmos evolutivos que usan estructuras de datos de árbol como genotipos. En segundo lugar, puede ser definido también como el conjunto de algoritmos evolutivos que evolucionan programas, algoritmos o construcciones similares (Heitkotter y Beasley, 1998; Koza, 1992b).

En los algoritmos evolutivos simples, la estructura de datos que almacena un miembro de la evolución de la población de soluciones es de tamaño fijo. Esto significa que se debe tener cuidado en escribir una estructura de datos que sea lo suficientemente general para que tenga el potencial de contener la solución. En los problemas de optimización, el tener una estructura general puede ser un gran problema. Una solución a este problema fue inventada por John Koza y es llamada programación genética (Koza, 1992a, Kinner, 1994; Koza, 1994; Kinner y Angeline, 1996; Banzhaf et al., 1998; Koza, 1999; Ashlock, 2005).

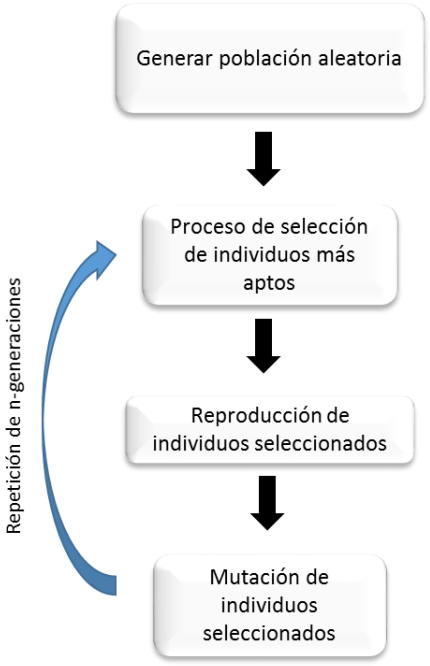

Figura 3.1: Esquema gráfico de la programación genética

En la PG se evoluciona una población de programas de computadora. La PG estocásticamente transforma la población en programas nuevos y mejores (en el mejor de los casos). El proceso estocástico jamás puede garantizar tener buenos resultados. La PG es en esencia un método aleatorio controlado, pero es capaz de no caer en trampas en donde un método determinístico se estancaría. Como en la naturaleza, la programación genética, es capaz de evolucionar conjuntos de programas y encontrar nuevas formas de resolver problemas. En la Figura 3.1 se muestra un esquema de la forma básica en que trabaja la programación genética.

(1) Generar una población inicial de programas aleatorios compuestos de funciones primitivas y terminales.
(2) Iterativamente realiza los siguientes procesos internos hasta que el criterio de finalización es satisfecho.
    a. Ejecuta cada programa de la población y asigna un valor de aptitud de acuerdo con qué tan bien resuelve se el problema.
    b. Crear una nueva población de programas usando los operadores genéticos. Estos operadores son aplicados a cierta población seleccionada con una probabilidad con base en su aptitud (mientras mejor aptitud tenga el programa, es más probable que sea seleccionado). Los operadores pueden ocurrir con cierta probabilidad.
        i. Recombinación o reproducción: Combina los genes de dos programas para crear dos descendientes que formarán parte de la nueva población.
        ii. Mutación: Altera genes de un individuo que formará parte de la nueva población.
(3) El mejor programa en la población producido durante la ejecución de la programación genética se tomará como la solución (o solución aproximada) del problema.

Algoritmo 3.1: Algoritmo de la programación genética.

La programación genética evoluciona poblaciones de programas con los pasos mostrados en el algoritmo 3.1.

3.1.1. Breve historia de la Programación Genética

La historia de la programación genética empieza con los primeros días de la ciencia de la computación (Angeline, 1998). Friedberg (1958) deja las primeras huellas en esta área al usar un algoritmo de aprendizaje que mejora en cada paso que da el programa. El programa fue representado como una secuencia de instrucciones para una computadora teórica llamada Herman (Friedberg et al., 1959). Friedberg (1959) no usó una evolución basada en poblaciones como medio para la búsqueda de programas. Esto fue debido a que la idea de algoritmos evolutivos no estaba desarrollada totalmente aún y también por la capacidad limitada de los ordenadores de esa época.

En esa época, Samuel (1959) aplica un algoritmo de aprendizaje máquina en un juego de damas, creando el primer programa en el mundo que cuenta con auto aprendizaje. En la sección de trabajo futuro de su artículo, sugiere que lo implementado en el juego de damas puede ser utilizado en problemas de regresión simbólica.

El uso de la programación evolutiva para evolucionar máquinas de estado finito fue desarrollado por Fogel et al. (1966), usando distintos tipos de mutación (sin recombinación) para crear una descendencia de individuos exitosos.

Catorce años después, la siguiente generación de científicos busca nuevas formas de evolucionar programas. Nuevos resultados fueron reportados por Smith (1980) en su tesis de doctorado. En otros estudios se evolucionaron árboles de expresiones booleanas para problemas de clasificación (Forsyth, 1981, 1989; Forsyth y Rada, 1986).

La década de los 80 fue un periodo muy productivo para el desarrollo de la PG. Cramer (1985) aplicó un algoritmo genético para evolucionar un programa escrito en un subconjunto del lenguaje PL en 1985. Este algoritmo genético usó una cadena de enteros como genotipo y empleó un mapeo de genotipo a fenotipo que lo transformaba recursivamente en estructuras de árbol. Mientras que Hicklin (1986) y Fujuki (1986)

implementaron operaciones de reproducción para manipular las claúsulas `if-then` en los programas en LISP.

La programación genética fue totalmente aceptada a finales de esta productiva década, principalmente por los trabajos Koza (1988, 1989). Se estudiaron también varias aplicaciones de la programación genética, tales como aprendizaje de funciones booleanas (Koza, 1990a, 1990d), el problema de la hormiga (Koza, 1990b, 1990c, 1992b), y problemas de regresión simbólica (Koza, 1990d, 1992b), que es un método para obtener expresiones matemáticas que ajusten una serie de puntos dados. Koza (1988, 1992c) formalizó y patentó la idea de emplear genomas puramente de estructuras de datos de árbol en lugar de cadenas de cromosomas como los que usan los algoritmos genéticos. En regresión simbólica, las estructuras de árbol pueden, por ejemplo, codificar expresiones LISP donde un nodo representa una operación matemática y sus nodos hijos son parámetros de la operación. Los nodos hoja son entonces símbolos terminales como números o variables. Con la PG no sólo se pueden codificar expresiones matemáticas, sino también se pueden representar programas más complejos (Sumathi et al. 2008).

3.1.2. Representación

En programación genética los programas se representan en forma de árbol más que como líneas de código. Por ejemplo, la Figura 3.2 muestra la representación de árbol del programa (/ (+ x y) (- z 4)), cabe decir que la programación genética se refiere a cada solución generada como un programa haciendo alusión, en el caso de nuestra investigación, a una expresión matemática (posible solución a nuestro ajuste de curvas). Las variables y constantes en el programa (x, y, z, 4) son hojas en el árbol que en PG son llamadas terminales, mientras que los operadores aritméticos (+, /, -) son nodos internos llamados funciones. Los conjuntos de funciones y terminales permitidos forman el conjunto primitivo de un sistema de programación genética. Es común en la programación genética representar expresiones en notación prefija, similar a como se usa en LISP. Por ejemplo, (+ (- 4 x) (/ 2 y)). Esta notación frecuentemente hace más fácil la relación entre expresiones y subexpresiones y sus correspondientes árboles y

subárboles. La forma en que uno puede implementar árboles en programación genética depende en gran medida del lenguaje y las bibliotecas que se usen. Los lenguajes que cuenten con recolección automática de basura y listas dinámicas como tipos de datos fundamentales, hacen más fácil implementar árboles de expresiones y las operaciones de PG necesarias. Los lenguajes más usados en el área de la inteligencia artificial como LISP y Prolog, además de ser lenguajes asociados con otras herramientas de programación científica como MATLAB y Mathematica tienen estas facilidades. En otros lenguajes uno puede verse en la necesidad de crear listas y árboles o usar bibliotecas que generen dichas estructuras de datos.

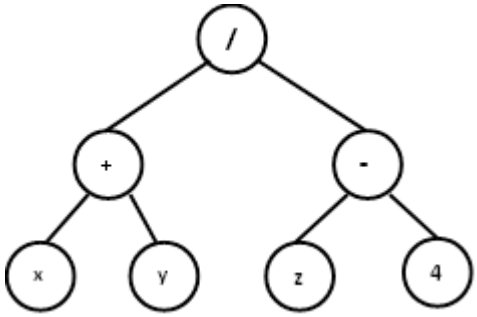

Figura 3.2: Estructura de árbol representando el programa (/ (+ x y) (- z 4)).

3.1.3. Inicialización de la población

Como en otros algoritmos evolutivos, en programación genética los individuos de la población inicial son típicamente generados aleatoriamente. Hay varias formas de generar esta población inicial. Describiremos dos métodos (completo y expansivo) y una combinación ampliamente usada de los dos, conocida como *mitad-y-mitad*.

En ambos métodos tanto el método completo como el método expansivo, los individuos iniciales son generados de tal forma que no excedan la profundidad máxima especificada por el usuario. La profundidad de un nodo es el número de aristas que necesitan ser recorridas, empezando desde el nodo raíz (del cual se asume que tiene profundidad cero), para llegar al nodo en cuestión. La profundidad del árbol será la de la hoja más profunda.

El método completo es llamado así debido a que se generan árboles completos, es decir, todas las hojas son de la profundidad máxima especificada. La Figura 3.3 muestra la construcción de un árbol con éste método.

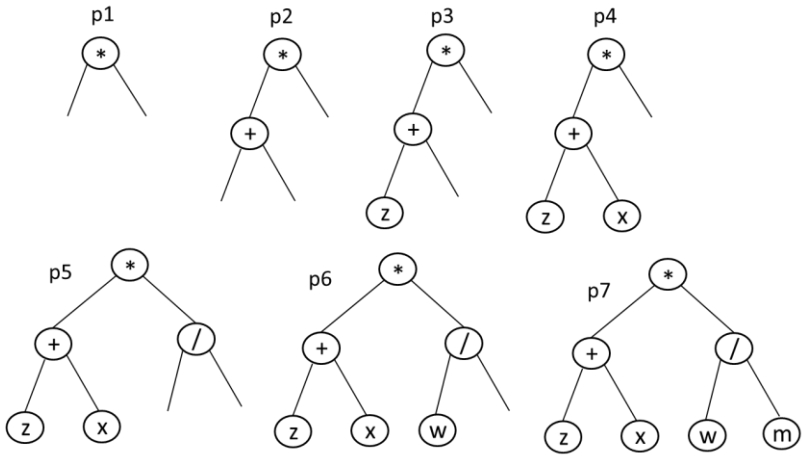

Figura 3.3: Pasos para construir un árbol de profundidad 2, usando el método completo

Aunque el método completo genera árboles donde las hojas tienen la misma profundidad esto no quiere decir que todos los árboles tengan la misma forma o cuenten con el mismo número de nodos (tamaño de árbol). Esto sólo sucede cuando todas las funciones en el conjunto primitivo tienen la misma aridad, es decir, requieren el mismo número de operandos. Sin embargo, incluso cuando es usado un conjunto primitivo con aridad mixta, las formas y tamaños producidos pueden ser bastante limitados.

El método expansivo, por el contrario, permite la creación de árboles con más variedad de formas y tamaños. Los nodos son seleccionados a partir de todo el conjunto primitivo, tanto funciones como terminales, hasta que la profundidad máxima es alcanzada. Como se puede elegir en cada paso de la construcción del árbol tanto un terminal como un operador, entonces ciertos niveles del árbol pueden quedar truncados por un terminal, aunque la profundidad no sea la máxima establecida; esto se puede observar en la Figura 3.4.

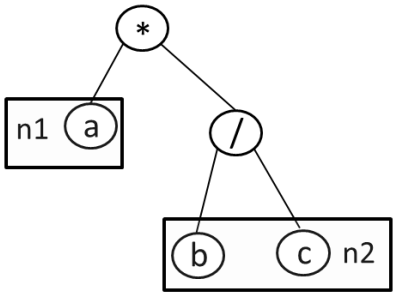

Figura 3.4: Árbol de profundidad 2, construido con el método expansivo

Debido a que los métodos completo y expansivo por sí mismos no generan un amplia gama de formas y tamaños de árbol, Koza en 1992 propone el método *mitad-y-mitad* que consiste en crear la mitad de la población con el método expansivo y la otra mitad con el método completo, con el único fin de crear una población más diversa.

3.1.4. Selección de la población

Tal como en la mayoría de los algoritmos evolutivos, los operadores genéticos son aplicados a individuos que son probabilísticamente seleccionados basados en su aptitud. Esto significa que los individuos mejor adaptados tienen más probabilidad de tener más cantidad de programas hijos que los individuos poco aptos. El método empleado más común para seleccionar individuos el de selección por torneo.

En la selección por torneo, un número de individuos son seleccionados de la población. Estos son comparados entre ellos y se escoge el mejor para ser padre. Para realizar la operación de cruce o recombinación se necesitan dos padres. En consecuencia el proceso se repite dos veces. Notemos que la selección por torneo sólo reconoce cuál programa es mejor que los otros y no necesita saber qué tan bueno es. Esto evita que un individuo demasiado bueno plague con su descendencia la siguiente generación. Si esto ocurriera se produciría una pérdida grave de la diversidad estropeando el espacio de búsqueda. Al usar este método permitimos que individuos con poca aptitud tengan oportunidad dentro de un grupo más reducido de ser seleccionados para ser padres y transmitir sus genes a la siguiente generación. Existen numerosas propuestas de métodos de selección muy

bien documentados que se encuentran en la literatura. Por cuestiones prácticas revisaremos sólo dos métodos de selección: selección por ruleta y selección por torneo binario.

3.1.4.1. Selección por ruleta

Esta técnica fue propuesta por De Jong (1975) y ha sido el método más comúnmente usado desde los orígenes de los algoritmos genéticos. El algoritmo es simple, pero ineficiente y su complejidad es $O(n^2)$. La selección por ruleta consiste literalmente en girar una ruleta en donde se encuentran los individuos de la población ocupando un área de la ruleta. Dependiendo de su aptitud, el individuo ocupará más o menos área de la ruleta teniendo a su vez mayor o menor probabilidad de ser seleccionado.

Usando este tipo de selección podríamos tener el caso en el que el individuo menos apto pueda ser seleccionado una o varias veces, lo cual puede ser un punto de debate. Sin embargo, si siempre seleccionamos a los mejores individuos, perdemos diversidad en las poblaciones y podríamos tener una convergencia prematura, estancándonos en un máximo local (si damos por hecho de que se trata de un problema de maximización).

El algoritmo de la ruleta es el siguiente:

- Calcular la suma de valores esperados SVE.
- Repetir N veces (N es el tamaño de la población).
    - Generar un número aleatorio NA entre 0.0 y SVE.
    - Ciclar a través de los individuos de la población sumando los valores esperados hasta que la suma sea mayor o igual a NA.
    - El individuo que haga que esta suma exceda el límite será el individuo seleccionado.

A continuación veremos un ejemplo:

Individuos	Aptitud	Valor esperado
1	25	0.35
2	81	1.13
3	36	0.51
4	144	2.01
	$\sum = 286$	$\sum = 4.0$

$f_{total} = \frac{286}{4} = 71.5$	$v_{e_i} = \frac{f_i}{f_{total}}$

$$SVE = Suma\ de\ valores\ esperados$$

- Se genera un número aleatorio, NA que pertenece a [0.0 , SVE]
- El número generado es NA = 1.3
- La suma de valores esperados hasta el individuo 1 es 0.35, por tanto, no es seleccionado.
- La suma de valores esperados hasta el individuo dos es 1.48 (0.35 + 1.13), mayor que el número aleatorio, por tanto es el individuo que rebasa el límite y es seleccionado.

3.1.4.2. Selección por torneo binario

Esta técnica fue propuesta por Wetzel (1983). La selección por torneo binario es muy usada debido a su bajo consumo de recursos puesto que tiene complejidad O(*n*).

La idea básica del método es seleccionar con base en las comparaciones directas de los individuos. Hay dos versiones de la selección mediante torneo: determinista y probabilística.

El algoritmo de la versión determinista es el siguiente:

- Mezclar aleatoriamente los individuos de la población.

- Escoger un número *i* de individuos (comúnmente 2).
- Compararlos con base en su aptitud.
- El ganador del torneo es el individuo más apto.
- Debe barajarse la población un total de *i* veces para seleccionar *n* padres, donde *n* es el tamaño de la población.

A continuación veamos un ejemplo:

Individuo en orden	**Aptitud**	**Reorganización**	**Ganadores**
1	254	2	Entre el individuo 2 y 6, gana 6 por tener más aptitud
2	47	6	
3	457	1	Entre el individuo 1 y 3, gana 3 por tener más aptitud
4	194	3	
5	85	5	Entre el individuo 5 y 4, gana 4 por tener más aptitud
6	310	4	

Reorganización	**Ganadores**
4	1
1	
6	6
5	
2	3
3	

Por tanto los padres seleccionados son: 6 y 1, 3 y 6, 4 y 3. Esta es una forma de hacerlo, sin embargo, el programador puede improvisar.

El algoritmo de la versión probabilística es igual al anterior, excepto por el paso donde se escoge al ganador. En este caso no siempre se selecciona al individuo con aptitud más alta, en cambio, se recurre a cierta probabilidad (comúnmente una probabilidad muy

baja) en la que el individuo menos apto pueda ser seleccionado con el fin de tener diversidad en la población.

A diferencia de la complejidad O(n^2) del método de selección ruleta, este método tiene complejidad O(n) y es usado cuando el número de individuos en las poblaciones es demasiado alto.

3.1.5. Recombinación y mutación

Básicamente los algoritmos evolutivos mantienen una población de estructuras que evolucionan de acuerdo a las reglas de selección y de otros operadores tales como recombinación y mutación (Spears, 1998). La forma más común de cruce o recombinación es el cruce de subárboles. Dados dos padres, se selecciona un punto de cruce aleatorio (un nodo), se copia el sub árbol generado, tomando como nodo raíz el nodo que se seleccionó previamente. El mismo proceso se realiza con el segundo padre y cada subárbol es insertado en el padre contrario en el punto de cruce antes seleccionado. El proceso se muestra en la Figura 3.5. Las copias son usadas para no destruir al individuo original; de esta forma si éste es seleccionado múltiples veces, puede formar parte de todas las descendencias que se generen.

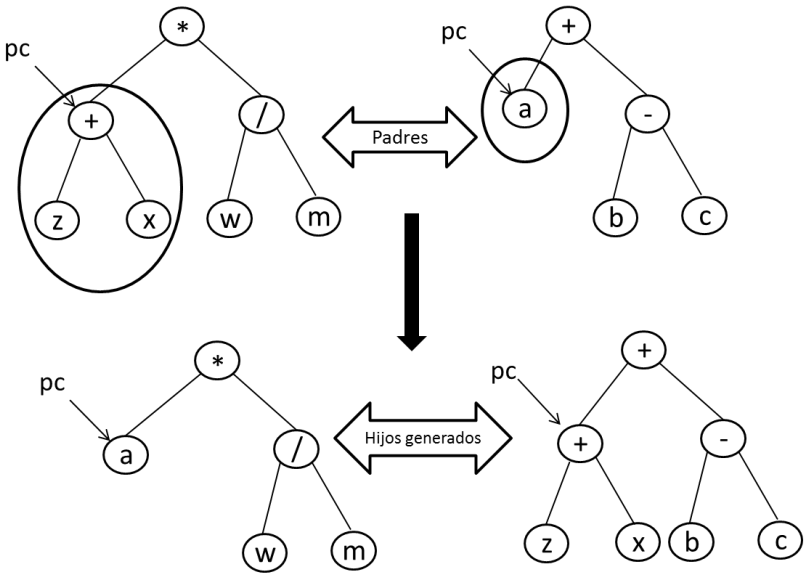

Figura 3.5: Aplicación de la recombinación. (pc = punto de cruce)

La forma más común de mutación en programación genética es muy parecida a la forma en que se realiza la recombinación. Se selecciona un punto aleatorio dentro del árbol y se cambia el subárbol por un nuevo subárbol generado aleatoriamente (ver Figura 3.6). Este método de mutación se puede ver como el cruce entre un programa y un nuevo programa aleatorio. Esta operación es conocida como mutación por subárbol (Kinnear y Angeline, 1996).

Otra forma común de mutación es "mutación por punto" que es equivalente al cambio de bit que se usa en algoritmos genéticos (Goldberg, 1989). En la mutación por punto un nodo aleatorio es seleccionado y la primitiva que se encuentre en ese nodo es reemplazada por otra primitiva aleatoria de la misma aridad seleccionada del conjunto de primitivas.

Cuando el método de mutación por subárbol es aplicado, la mutación se aplica sólo a un subárbol del individuo. Por otra parte al aplicar el método de mutación por punto, se evaluará una probabilidad por cada nodo de cada individuo de la población.

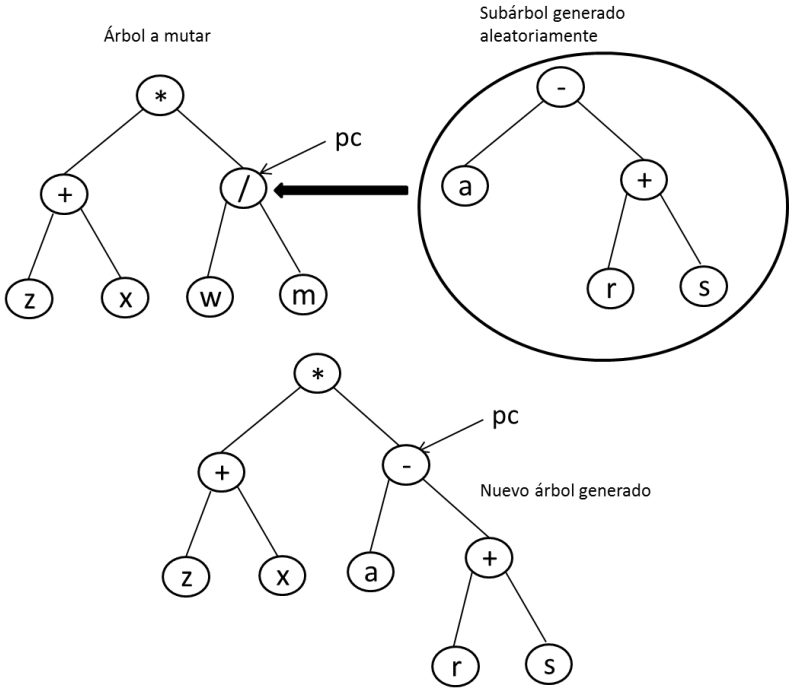

Figura 3.6: Aplicación de mutación por subárbol. (pc = punto de cruce)

Típicamente, la recombinación es aplicada con una probabilidad alta (del 90% o más), en contraparte con la mutación que se aplica con una probabilidad cercana al 1%.

3.1.6. Especificaciones importantes de la programación genética

A la hora de implementar un sistema de programación genética hay que decidir ciertas cuestiones significativas que nos van a permitir poner en marcha nuestro trabajo. Estas cuestiones preliminares son:

1. El conjunto de terminales;
2. El conjunto de funciones;
3. La medida de aptitud;
4. Los parámetros que se usaran en la ejecución;
5. El criterio de terminación de programa.

3.1.6.1. El conjunto de terminales

Mientras que es usual usar la frase *evolución de programas* cuando describimos un sistema de programación genética, éste no suele evolucionar programas con instrucciones en lenguajes de programación que los humanos usamos para desarrollar software. Es más común utilizarlo para evolucionar expresiones y fórmulas en un lenguaje más específico y reducido. Este mal entendido fue ocasionado porque en un principio se ambicionó demasiado sobre los alcances de la programación genética. En estos días el poder de cómputo ha evolucionado exponencialmente. Sin embargo, aún es costoso tener una supercomputadora para probar algoritmos como éstos.

Los primeros dos pasos importantes en un sistema de programación genética son la definición del conjunto de terminales y de funciones. Esto es, juntos definen los ingredientes que estarán disponibles para que el sistema de PG pueda crear los programas.

El conjunto de terminales puede estar formado por:

- **Entradas externas del programa.** Éstas típicamente son variables (ejemplo: x, y, z).
- **Funciones sin argumentos.** Éstas pueden ser incluidas porque regresan distintos valores cada vez que son usadas, tal como la función *rand(),* la cual regresa un número aleatorio, o la función *dist_to_wall()* que regresa la distancia de un obstáculo de un robot que está siendo controlado por un sistema de PG. Otra posible razón es porque la función produce efectos secundarios. Las funciones con efectos secundarios hacen más que sólo regresar un valor, ellas tal vez podrían cambiar estructuras de datos globales, escribir o dibujar algo en pantalla, controlar motores de un robot, etc.
- **Constantes.** Pueden ser especificadas previamente, creadas aleatoriamente como parte del proceso de creación de árboles o creadas en el proceso de mutación.

Usar una primitiva como *rand()* puede aportar la característica a un programa de entregar un resultado distinto cada vez que sea llamado, incluso si éste tiene las mismas entradas. Esto puede ser útil en algunas aplicaciones. Sin embargo, más frecuentemente requerimos un conjunto de constantes aleatorias fijas que sean generadas como parte del proceso de inicialización de la población. Esto típicamente se cumple introduciendo un terminal que representa una constante efímera. Cada vez que este terminal es elegido en la construcción de un árbol inicial (o en la creación de un subárbol creado en la operación de mutación), un valor aleatorio diferente es generado y utilizado para ese terminal en particular. Este permanecerá fijo el resto de la ejecución. El uso de constantes efímeras aleatorias es típicamente denotado por el símbolo Ŕ en el conjunto de terminales.

3.1.6.2. El Conjunto de funciones

El conjunto de funciones en un sistema de programación genética es típicamente especificado por la naturaleza del problema que se esté intentando resolver. En un problema numérico, por ejemplo, el conjunto de funciones podría consistir en funciones aritméticas como +, -, * y /. Sin embargo, todo tipo de funciones y construcciones

pueden ser usadas. La Tabla 3.1 muestra un ejemplo de algunas de las funciones que se encuentran en la literatura.

Algunas veces el conjunto primitivo incluye funciones y terminales especializados diseñados para resolver problemas de dominio específico. Por ejemplo, si el objetivo es programar un robot que se dedique a limpiar pisos, entonces, el conjunto de funciones puede incluir acciones como mover-izq, mover-der y girar.

Tabla 3.1: Ejemplo de conjunto primitivo (conjunto de funciones y conjunto de terminales).

Conjunto de funciones	
Tipo de primitiva	***Ejemplo***
Aritmética	*, +, -
Matemática	sin, cos, exp
Booleana	and, or, not
Condicional	if-then-else
Ciclo	for, repeat

Conjunto de terminales	
Tipo de primitiva	***Ejemplo***
Variables	x, y
Valores constantes	9, 8.1, 3.2
funciones de aridad 0	rand(), mover-izq

3.1.6.2.1. Propiedad de cierre

Para que un sistema de programación genética trabaje correctamente, es requisito que el conjunto de funciones cumpla con la propiedad conocida como cierre (Koza, 1992a), que a su vez se puede dividir en las propiedades de consistencia de tipo y de evaluación segura.

La consistencia de tipo es requerida porque el proceso de recombinación puede mezclar y unir individuos de forma arbitraria. Como consecuencia es necesario que cualquier subárbol pueda ser usado en cualquier posición dentro de otro árbol, porque es posible que dentro de un proceso de mutación se genere cualquier estructura. Por lo tanto, es

requisito que todas la funciones tengan consistencia de tipo, pues esto significará que todos los valores que regresen serán del mismo tipo y que cada uno de sus argumentos también serán de ese tipo. Por ejemplo, las funciones aritméticas +, -, *, / pueden ser definidas como que cada una de ellas recibe dos argumentos de tipo numérico real y devuelve un valor de tipo real también. Algunas veces la consistencia de tipo puede ser débil por algún mecanismo de conversión automática de tipo. Podemos, por ejemplo, convertir un valor numérico a un valor booleano asignándole un valor de falso a todos los números negativos y un valor verdadero a los positivos. Sin embargo, los mecanismos de conversión pueden producir sesgos inesperados en el proceso de búsqueda por lo que deben ser usados cuidadosamente.

La propiedad de consistencia de tipo puede parecer bastante limitante pero muchas veces una simple reestructuración de las funciones puede resolver problemas que pueden llegar a ocurrir. Por ejemplo, una función "`if`" frecuentemente toma tres argumentos: la evaluación, el valor que regresará si la evaluación es verdadera y el valor que regresará si la evaluación es falsa. El primero de los tres argumentos es claramente de tipo booleano lo cual sugiere que no puede ser usado con funciones numéricas como "+". Esto puede ser tratado fácilmente convirtiendo valores numéricos en booleanos como se dijo arriba. Alternativamente, podríamos reemplazar las tres entradas de la función "`if`" por cuatro entradas (a, b, c, d). Entonces se implementaría: "`if a < b` entonces regresa c, si no, regresa d".

La propiedad de consistencia de tipo es importante en los procesos de recombinación y mutación para que dos árboles padre no generen descendencia errónea.

El otro componente de la propiedad de cierre es la **evaluación segura**. La evaluación segura es importante porque las funciones generadas pueden causar errores en tiempo de ejecución. Una función evolucionada podría provocar una división por cero o llamar una función mover-adelante cuando el robot está frente a la pared o frente a un precipicio. Es común usar versiones protegidas de funciones numéricas que puedan llegar a generar excepciones tal como la división, el logaritmo, la potencia, la raíz cuadrada, etc. La versión protegida de una función primero evalúa los problemas potenciales que pudieran presentarse con las entradas, para después ejecutar la instrucción correspondiente; si un

problema se llegara a presentar, entonces, algún valor será devuelto por defecto. La división protegida (frecuentemente denotada por %) verifica si su segundo argumento es cero. Si se da ese caso, la división protegida devolverá el valor 1, independientemente del valor que tenga el primer argumento). De forma similar, en un programa para un robot, una operación de mover-adelante puede ser modificada por hacer-nada si el movimiento generará algún daño al robot.

Una alternativa para proteger las funciones es atrapar las excepciones en tiempo de ejecución y reducir drásticamente la aptitud del programa que genere tales errores. Sin embargo, si la probabilidad de generar expresiones erróneas es bastante alta, esto puede dejar demasiados programas con aptitud parecida y muy baja. Esto hace difícil el proceso de selección para elegir cuales podrían ser los mejores padres.

Un tipo de error en tiempo de ejecución que es muy difícil de verificar, es un desbordamiento numérico. Sin embargo, es común en muchos lenguajes de programación ignorar el desbordamiento y simplemente redondear el valor. Si esto es inaceptable, entonces la implementación del sistema de programación genética deberá tener las evaluaciones correspondientes para capturar estos errores y hacer la corrección pertinente.

3.1.6.2.2. Propiedad de suficiencia

Hay una propiedad más que los conjuntos primitivos deben cumplir y es la suficiencia. La suficiencia significa que es posible expresar la solución al problema en cuestión, usando los elementos del conjunto primitivo. Desafortunadamente, la suficiencia sólo puede ser garantizada para aquellos problemas donde la teoría o la experiencia con otros métodos nos dice que la solución puede ser generada por medio de combinar los elementos del conjunto primitivo.

Como ejemplo de suficiencia observemos el siguiente conjunto primitivo {and, or, not, x_1, x_2, …, x_n}. Este conjunto siempre será suficiente para problemas de inducción booleana, por el simple hecho que puede producir todas las funciones booleanas de las variables x_1, x_2, …, x_n. Un ejemplo de insuficiencia es el conjunto {+, -, *, /, x, 0, 1, 2,

3}, el cual es incapaz de representar funciones trascendentes. La función exp(x), por ejemplo, es trascendente y no puede ser representada con la combinación de {+, -, *, /, x, 0, 1, 2, 3}. Cuando el conjunto primitivo es insuficiente, la programación genética sólo puede, en el mejor de los casos, desarrollar programas aproximados a la solución deseada.

Sin embargo, en muchos casos una aproximación puede ser bastante eficiente para el propósito del usuario. Añadir unas pocas primitivas no necesarias con el objetivo de lograr la suficiencia, no afecta demasiado el rendimiento del sistema, aunque hay casos en los que se puede sesgar el sistema en forma inesperada.

3.1.6.3. La función de aptitud

Un problema de optimización puede ser definido mediante la especificación el conjunto de los candidatos viables y una medida de evaluación de su valor (Pelikan, 2002). En los primeros pasos preliminares se define el conjunto primitivo, y por lo tanto, implícitamente se define el espacio de búsqueda, el cual va a explorar el sistema de PG. Esto incluye todos los programas que serán construidos por medio de la composición de las primitivas en todas las posibles formas. Sin embargo, hasta este momento, nosotros no sabemos cuáles elementos o regiones de este espacio de búsqueda son los idóneos, es decir, cuáles regiones del espacio incluyen programas que resuelven o aproximadamente resuelven el problema en cuestión. Este es el objetivo de la función de aptitud, la cual es el mecanismo primario para dar una declaración de los requisitos primarios del sistema de PG. Por ejemplo, supongamos que el objetivo es obtener automáticamente la síntesis de un amplificador de señal. Entonces, la función de aptitud es el mecanismo que le dice al sistema de PG cuál es el circuito que amplifica de la mejor manera una señal de entrada.

La aptitud puede ser medida de distintas formas. Por ejemplo, en términos de la cantidad de error entre la salida del programa y la salida deseada; de la cantidad de tiempo (gasolina, dinero, etc.) requerida para que el programa llegue al estado deseado; de la precisión del programa para reconocer patrones o clasificar objetos.

Hay algo inusual que tienen las funciones de aptitud de la PG que las diferencian de otras usadas en otros algoritmos evolutivos. Debido a que las estructuras que son evolucionadas en PG son programas, la evaluación de la aptitud normalmente requiere que se ejecuten muchas veces todos los programas en la población. Así que, es muy común usar un intérprete para evaluar los programas evolucionados.

```
1: if expr es lista then
2:       proc = expr(1) {No-terminal: extrae-raíz}
3:       if proc es función then
4:          valor = proc( eval(expr(2)), eval(expr(3)), ... ) {Función: evaluar argumentos}
5:       else
6:          valor = proc( expr(2), expr(3), ...) {Macro: no evaluar argumentos}
7:       end if
8: else
9:       if expr es variable or expr es constante then
10:         valor = expr {Terminal variable o constante: leer el valor}
11:      else
12:         valor = expr() {Terminal de aridad 0: ejecuta función con efecto secundario}
13:      end if
14: end if
15: return valor
```

Algoritmo 3.2: Pseudocódigo de un intérprete para un sistema de PG.

Interpretar un programa (una estructura de árbol) significa ejecutar los nodos del árbol en un orden que garantice la evaluación correcta del programa. Esto usualmente se hace recorriendo el árbol recursivamente empezando por el nodo raíz y posponiendo la evaluación de cada nodo hasta que los valores de sus hijos (argumentos) sean conocidos. Diferentes órdenes para recorrer los árboles son conocidas, tal como ir de las hojas hasta la raíz. Si ninguna primitiva tiene efectos secundarios, ambas formas son equivalentes. El proceso recursivo llamado “profundidad-primero” se muestra en la Figura 3.7. El algoritmo 3.2 muestra el pseudocódigo del proceso de interpretación. El código asume que los programas son representados como expresiones en notación prefija y que tales expresiones pueden ser tratadas como listas de componentes.

En algunos problemas estaremos interesados en la salida producida por el programa, llamado valor de retorno, cuando evaluamos el árbol empezando en el nodo raíz. En otros casos podemos estar interesados en las acciones realizadas por un programa compuesto de funciones con efectos secundarios. En cualquier caso, la aptitud de un programa típicamente depende de los resultados producidos de su ejecución con diferentes entradas o bajo una variedad de distintas condiciones. Por ejemplo, el programa puede ser probado con las posibles combinaciones de las entradas x_1, x_2, …, x_n. Alternativamente, el programa de control de un robot podría ser probado con varias locaciones de inicio. Estos diferentes casos de prueba comúnmente contribuyen a que el valor de aptitud de un individuo se incremente y por eso son llamados casos de aptitud.

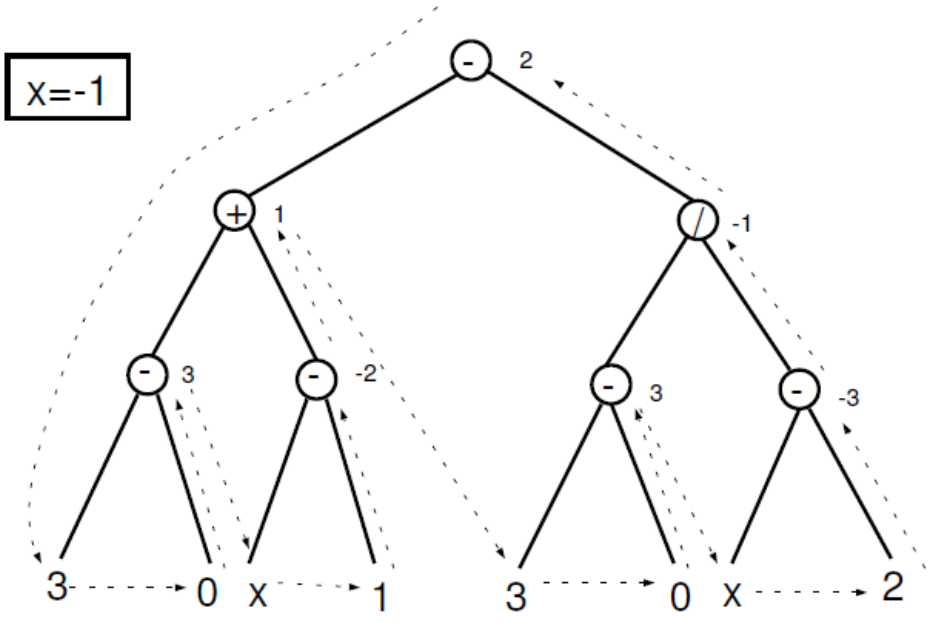

Figura 3.7: Ejemplo de interpretación de la sintaxis de un árbol

Otra característica común de la medida de aptitud en la PG es que para muchos problemas prácticos hay un multiobjetivo, es decir, se combinan dos o más elementos diferentes que están frecuentemente en competencia entre ellos. El área de optimización multiobjetivo generalmente es un área de investigación en PG y en aprendizaje automático. Los problemas con múltiples objetivos surgen de manera natural en la

mayoría de las disciplinas y su solución ha sido un reto para los investigadores durante mucho tiempo (Coello et al., 2007).

3.1.6.4. Los parámetros de la programación genética

El cuarto paso preliminar trata sobre los parámetros de control de la ejecución. El parámetro de control más importante es el tamaño de la población (Coello et al., 2007). Otros parámetros de control incluyen la probabilidad de aplicación de los operadores genéticos, el tamaño máximo de los programas y otros detalles de la ejecución.

Es imposible hacer recomendaciones generales sobre qué valores son los óptimos para los parámetros de control, pues esto depende mucho de los detalles de la aplicación que se esté implementando. Sin embargo, la programación genética es una aplicación robusta en la práctica, por lo que es probable que diferentes valores en los parámetros de control trabajen bien. Como consecuente, uno no necesita gastar mucho tiempo tratando de ajustar esos valores.

Es común crear la población inicial aleatoriamente usando el método *mitad-y-mitad* con un rango de profundidad de 2 a 6 niveles. El tamaño inicial de los árboles depende del número de funciones, el número de terminales y la aridad de las funciones. Sin embargo, el proceso evolutivo moverá rápidamente a la población de su estado inicial.

Tradicionalmente el 90% de la descendencia es creada por el proceso de recombinación, pero por otra parte la creación por recombinación y la mutación del 50% - 50% han dado buenos resultados.

En muchos casos, la principal limitante del tamaño de la población es el tiempo que toma evaluar las aptitudes y no el espacio que se utiliza para almacenar los individuos. Como regla, se prefiere tener una población lo más grande posible, que el equipo donde se ejecute pueda manejarla sin problemas; normalmente el tamaño de la población debería ser de al menos 500, pero la mayoría de la gente usa poblaciones mucho más grandes.

Normalmente, el número de generaciones está limitado entre diez y cincuenta; la búsqueda más productiva es usualmente realizada en ese rango de generaciones y si una solución no ha sido encontrada hasta ese momento, entonces es poco probable que sea encontrada más adelante. Lo más popular entre la gente es hacer que el tamaño de la población sea tan grande como se pueda, pero hay algunos que sugieren hacer más pequeña la población y más grande el número de generaciones.

Algunas aplicaciones no requieren especificar un tamaño máximo de los árboles. Sin embargo, debido al crecimiento incontrolado que se puede dar en los individuos (llamado hinchamiento) es muy común poner límite al crecimiento de los mismos. El crecimiento descontrolado es un problema serio en programación genética (Bosman, 2003).

3.1.6.5. El criterio de finalización de la ejecución

El quinto paso preliminar es especificar un criterio de finalización y el método de elección del resultado de la ejecución. El criterio de finalización puede ser un número máximo de generaciones.

Comúnmente se entrega la mejor solución encontrada en toda la población, pero en algunos casos se requiere un conjunto de los mejores resultados de la población evolucionada.

3.1.7. La evolución con estructuras diferentes

Hay muchos problemas en los que las soluciones no pueden ser modeladas directamente como programas de ordenador. Por ejemplo, en muchos problemas de diseño la solución es un artefacto de algún tipo: un puente, un circuito, una antena, una lente, etc. La programación genética ha sido aplicada a este tipo de problemas implementando ciertas modificaciones al sistema. El conjunto primitivo está configurado de tal manera que construya soluciones al problema. Por ejemplo, si el objetivo es la creación automática de un controlador electrónico para una planta, el conjunto de funciones podría incluir

componentes comunes como un circuito integrador, un circuito diferenciador y un circuito de ganancia, mientras que el conjunto de terminales podría ser una señal, una referencia y una salida de la planta. Cuando es elegida cada una de estas primitivas, entonces, inserta el dispositivo correspondiente dentro del controlador que se está construyendo. Si por otra parte, el objetivo es sintetizar circuitos electrónicos analógicos, el conjunto de funciones podría incluir componentes como transistores, capacitores, resistores, etc.

3.2. Conceptos básicos de ajuste de curvas con polinomios

El ajuste de curvas consiste en encontrar una curva que contenga una serie de puntos y que posiblemente cumpla una serie de restricciones adicionales. A cada punto que se intenta ajustar le podemos llamar restricción.

Comenzamos con una ecuación polinómica de primer grado, por ejemplo: $y = ax + b$. Esta línea tiene pendiente a. Sabemos que habrá una línea conectando dos puntos cualesquiera. Por tanto, una ecuación polinómica de primer grado es un ajuste perfecto entre dos puntos. Si aumentamos el orden de la ecuación a la de un polinomio de segundo grado, por ejemplo: $y = ax^2 + bx + c$, esto ajustará exactamente tres puntos. Si aumentamos el orden de la ecuación a la de un polinomio de tercer grado, obtendríamos: $y = ax^3 + bx^2 + cx + d$, que se ajustaría a cuatro puntos.

Una forma más general de decirlo es que se ajustará exactamente a cuatro restricciones. Si tenemos más de n+1 restricciones (siendo n el grado del polinomio), aún podemos hacer pasar la curva polinómica por ellas. No es seguro que vaya a existir un ajuste exacto a todas ellas pero podría suceder, por ejemplo, en el caso de un polinomio de primer grado, que se ajuste a tres puntos colineales. Sin embargo, se necesita algún método para evaluar cada aproximación. El método de mínimos cuadrados es una manera de comparar las desviaciones.

Ahora bien, podríamos preguntarnos la razón de querer un ajuste aproximado, cuando podríamos simplemente aumentar el grado de la ecuación polinómica para obtener un

ajuste exacto. Existen varias razones, la más interesante es que inclusive si existe un ajuste exacto, no quiere decir necesariamente que podamos encontrarlo. Dependiendo del algoritmo que se use, podríamos encontrar un caso divergente, donde no se podría calcular el ajuste exacto, o el coste computacional de encontrar la solución podría ser muy alto. De cualquier modo, tendríamos que acabar aceptando una solución aproximada.

3.2.1. Ajuste de curvas por mínimos cuadrados

Mínimos cuadrados es una técnica de análisis numérico enmarcada dentro de la optimización matemática, en la que dados un conjunto de pares ordenados variable independiente y variable dependiente, y una familia de funciones, se intenta encontrar la función continua dentro de dicha familia, que mejor aproxime a los datos, de acuerdo a un criterio de mínimo error cuadrático.

En su forma más simple, se intenta minimizar la suma de cuadrados de las diferencias en las ordenadas entre los puntos generados por la función elegida y los correspondientes valores en los datos.

3.2.1.1. La fórmula de error cuadrático medio

La fórmula de error cuadrático medio es frecuentemente usada para medir la diferencia entre valores predichos por un modelo o un estimador y los valores observados. Las diferencias individuales son llamadas residuos. La fórmula de error cuadrático medio sirve para agregar las magnitudes de los errores medidos en varios tiempos en una sola medida.

En algunas disciplinas, la fórmula de error cuadrático medio es usada para comparar diferencias entre dos cosas que pueden variar. Por ejemplo, cuando medimos el promedio de diferencias entre dos series de tiempo x y x', la fórmula que se aplica es:

$$Error\ Cuadrático\ Medio = \sqrt{\frac{\sum_{t=1}^{n}(x_t - x'_t)^2}{n}}$$

3.3. Ajuste de curvas mediante programación genética

Como hemos visto antes, podemos ajustar una curva con algún polinomio de grado *n* de acuerdo al número de restricciones que tenga.

Para evaluar el ajuste, es decir, para obtener una medida para saber qué tan parecida es la curva que genera nuestra función con los datos que estamos ajustando, podemos utilizar la fórmula de error cuadrático medio.

Ahora bien, con base en los trabajos vistos en el estado del arte (capítulo II), queda claro que podemos formar expresiones usando programación genética para resolver problemas de regresión lineal utilizando los operadores aritméticos básicos (+, -, *, /). Entonces, podemos complementar estos operadores con el operador potencia, con el que sería relativamente sencillo crear cualquier polinomio de grado *n* (donde *n* es un número entero desde cero hasta el límite establecido).

Trabajando bajo el supuesto de que vamos a ajustar una curva por polinomios, entonces, podemos utilizar la programación genética como un método para encontrar funciones que aproximen a las restricciones que plantee el ajuste.

Claro, que al desarrollar un sistema de programación genética debemos tomar en cuenta que vamos a lidiar con algunos problemas como la creación de expresiones erróneas (por ejemplo, una división con divisor igual a cero), o de forma más general, con el crecimiento agresivo de los árboles, con el costo computacional que se requiere y sobre todo tenemos que saber que la PG es un método en esencia estocástico, por tanto, no asegura jamás que se obtendrá el resultado óptimo.

Si los ajustes que se pretenden realizar están al alcance de un método determinístico, es mejor resolverlos con alguno de ellos; no obstante, como ya se ha explicado, la PG ha logrado ser exitosa en la resolución de problemas demasiado complejos para los

métodos habituales, debido a que tienen un poder de exploración superior en espacios de búsqueda enormes.

La PG y otros algoritmos evolutivos tienen la ventaja de, en un principio, dar literalmente grandes saltos en el espacio de búsqueda (llamado comúnmente exploración), lo cual permite ir recolectando muestras en diferentes áreas del espacio y después aplicar los operadores genéticos sobre ellas para poder crear soluciones cada vez mejores (llamado comúnmente explotación). Apoyándonos en esta característica, la PG actúa de forma más "inteligente" (sin meternos en controversias de la definición de inteligencia) en espacios de búsqueda muy grandes.

CAPÍTULO IV

IMPLEMENTACIÓN DE "GP CURVE FITTING"

Contenido

Este capítulo explica a grandes rasgos los puntos importantes de cómo se creó el sistema de programación genética "GP Curve Fitting". Se describen puntos importantes como el conjunto primitivo, la función de aptitud, elitismo, etc. Además, se muestran los parámetros que se pueden ajustar en el sistema para la ejecución del mismo.

4.1. GP Curve Fitting

GP Curve Fitting es un proyecto desarrollado en el lenguaje de programación Allegro Common LISP. El objetivo del desarrollo del sistema es realizar ajuste de curvas sobre conjuntos de datos económicos, con el fin de generar un modelo matemático para su estudio y análisis posterior.

Para poder generar los modelos matemáticos optamos por usar programación genética. El sistema permite el uso de múltiples variables independientes que el usuario puede agregar, esto es, existen *n* variables independientes que pueden pertenecer al modelo que se va a construir (algún polinomio de grado máximo 9). Cada una de las variables puede ser o no insertada como algún término del polinomio (suponiendo que la expresión formada es un polinomio), o expresión matemática que se forme. Esto convierte el ajuste de curvas en un problema de optimización combinatoria. El espacio de búsqueda crece con cada variable agregada y si sumamos a eso el tener que localizar los coeficientes correctos para un buen ajuste, podemos inferir que un método determinista no funcionaría.

GP Curve Fitting aprovecha las cualidades de la programación genética para moverse en el espacio de búsqueda y encontrar, si bien no el modelo óptimo, sí un modelo cercano al óptimo para el ajuste de curvas.

4.2. El conjunto primitivo

Antes de empezar a crear estructuras de árbol para representar las expresiones matemáticas, debemos tener claro qué conjunto de operadores aritméticos y conjunto de terminales se usarían, los cuales componen el conjunto primitivo.

El razonamiento es simple: puesto que se partió del supuesto de que el ajuste de curvas se realizaría con polinomios, entonces, la misma estructura del polinomio será la que nos brinde dicha información. Comencemos con el conjunto de operadores. Los operadores aritméticos que se pueden requerir en la construcción de un polinomio son los

operadores básicos más el operador de potencia y su inverso el logaritmo. Por lo cual, el conjunto de operadores queda de la siguiente manera:

$$\text{Operadores} = \{+, -, *, \%, \exp[0\text{-}9], \log\}$$

Se deben considerar ciertas restricciones como la división protegida "%" para evitar errores cuando el divisor sea cero. También la potencia restringida a los números enteros de cero al nueve "exp[0-9]".

Una vez que ha quedado claro el conjunto de operadores, comencemos a definir el conjunto de terminales que desde el punto de vista de la representación en estructuras de árbol, ocuparán los nodos hoja.

Definir el conjunto de terminales es simple, puesto que un nodo hoja sólo puede ser ocupado por un terminal (una variable o una constante) y nunca por un operador. Tomando en cuenta lo anterior, el conjunto de terminales debe estar formado por las variables independientes que ingrese el usuario al sistema y también por constantes aleatorias. El conjunto queda de la siguiente manera:

$$\text{Terminales} = \{ \text{VI}, \text{NA} \},$$

donde VI son la variables independientes y NA una constante aleatoria.

Hemos terminado de definir el conjunto primitivo, desde ahora el sistema de programación genética podrá disponer de él para formar expresiones y con ello podrá ser creada la población.

Recordemos que es muy importante determinar correctamente el conjunto primitivo de acuerdo al problema que se esté deseando resolver. De otra forma, encontrar buenas soluciones sería más difícil.

4.3. La medida de aptitud

Una vez creada una población de soluciones, necesitamos una forma de identificar a los individuos sobresalientes. GP Curve Fitting usa la técnica de selección por torneo, de

complejidad $O(n)$ que agiliza el proceso, aunque también cuenta con la técnica de selección por ruleta (de complejidad $O(n^2)$).

La forma en que un individuo puede ser evaluado o comparado respecto a otros es por medio de su aptitud, medida que nos dice qué tan buena o qué tan mala es una solución. Para lograr asignar una aptitud a cada individuo, hemos usado la fórmula de error cuadrático medio, la cual nos sirve para saber cuán cercanos son los datos que se generaron con la función encontrada por GP Curve Fitting y los datos en sí.

Debido a que la fórmula nos regresa un promedio de distancias entre pares de puntos ($\{(y_{t1}, f(x)_{t1}), (y_{t2}, f(x)_{t2}), \dots, (y_{tn}, f(x)_{tn})\}$), entonces, estamos frente a un problema de minimización, puesto que tenemos que minimizar la distancia entre los puntos. Una vez calculada la fórmula para cada individuo, podemos comenzar con el proceso de selección.

4.4. Elitismo

El elitismo es un proceso muy importante en la programación genética. Sin elitismo no se asegura la convergencia del algoritmo.

GP Curve Fitting, después de generar la población inicial y asignarle su debida aptitud a los individuos, recluta un grupo de *n* individuos élite (donde *n* es igual al 0.2% de la población total), es decir, escoge las *n* mejores soluciones de la población. Las soluciones élite son almacenadas en otra tabla para que no sufran cambios durante el proceso de reproducción y mutación en la generación g_m. Al iniciar la generación g_{m+1}, antes de realizar cualquier proceso, las soluciones élite son insertadas.

La tabla de soluciones élite siempre conserva el mismo número de individuos (*n* individuos). Si en futuras generaciones se consigue un individuo mejor que alguno en la tabla de soluciones élite, éste tomará su lugar.

Se da el caso que una generación empeore, al pasar por los procesos de reproducción y mutación, en lugar de mejorar. Al insertar a los individuos élite tenemos asegurado que jamás g_{m+1} será peor que g_m.

4.5. Parámetros del algoritmo

Un sistema de programación genética requiere que se ajusten ciertos parámetros antes de iniciar los experimentos.

GP Curve Fitting tiene parámetros establecidos por defecto, pues estos parámetros dieron mejores resultados en la etapa de prueba.

La recombinación es uno de los operadores genéticos que se aplica sobre dos individuos de la población, con base en una probabilidad. Esta probabilidad es de 0.9 y está representada por la variable global *probabilidad-cruza*.

La mutación es otro operador genético y de igual forma que la recombinación, necesita una probabilidad que determina si será aplicado o no. Esta probabilidad es de 0.01 y está representada por la variable global *probabilidad-mutación*.

La profundidad de los árboles está establecida en 6 niveles con la variable global *profundidad-árbol*.

Por último, el número de individuos élite está puesto en 0.2% de la población total usando la variable *mismo-élite*.

Por supuesto cualquier variable puede ser modificada para experimentos futuros.

4.6. Operadores de recombinación y mutación

La reproducción y la mutación (llamados operadores genéticos) son los operadores usados en la programación genética para evolucionar la población, los cuales son aplicados con cierta probabilidad. La literatura recomienda una probabilidad alta para la recombinación (de 0.9 o mayor) y una probabilidad baja para la mutación (de 0.1 o menor). El mejor ajuste de estos parámetros sólo se puede obtener a base de prueba y error, es decir, con la experimentación.

El método que usa GP Curve Fitting para aplicar el operador de recombinación se describe a continuación:

- Seleccionar dos individuos de la población de forma aleatoria (i_1, i_2)
- Seleccionar un punto aleatorio p_1 del individuo i_1.
- Seleccionar un punto aleatorio p_2 del individuo i_2.
- Guardar el subárbol generado desde p_1 en Sp_1.
- Eliminar el subárbol generado desde p_1 del individuo i_1 insertando en su lugar el subárbol generado desde el punto p_2 del individuo i_2.
- Eliminar el subárbol generado desde p_2 del individuo i_2 e insertar en su lugar el subárbol almacenado en Sp_1.

Si bien el operador de reproducción ayuda en la exploración del espacio de búsqueda, necesitamos un operador para la explotación. El operador de mutación nos permite hacer cambios muy pequeños a los individuos, a diferencia del operador de recombinación que aplica cambios significativos a los individuos. Esto se puede ver también, como saltos grandes en el espacio de búsqueda al aplicar la recombinación y saltos pequeños al aplicar la mutación, aunque no estrictamente. Un cambio en un operador aritmético en una expresión matemática ocasionado por una mutación puede arrojar un resultado totalmente diferente al de la expresión original. Sin hacer de este punto un debate, puesto que el objetivo de esta investigación no es una crítica de los mecanismos de la programación genética, comencemos a explicar la forma en la que opera la mutación en GP Curve Fitting.

Existen dos técnicas de mutación comunes. Una de ellas consiste en aplicar cambios a los nodos, por ejemplo, convertir un "+" en "-". Como sabemos la mutación requiere una probabilidad que le dice al algoritmo si debe aplicar el operador genético o no (no confundir los operadores genéticos de la programación genética con los operadores aritméticos de las expresiones matemáticas). En el caso de este tipo de mutación, la probabilidad es evaluada con cada nodo del árbol. GP Curve Fitting cuenta con este tipo de mutación con una probabilidad preestablecida de 0.01, lo que nos dice que cada nodo de cada árbol tiene 0.01 de probabilidad de ser mutado.

Otra forma de aplicar la mutación es modificar todo un subárbol del individuo; para esto se debe seleccionar un nodo aleatorio dentro del árbol. Todos los nodos hijo del nodo

seleccionado se eliminan y en su lugar queda un nuevo subárbol que se crea en ese instante. Ahora la probabilidad se evalúa por individuo y no por cada nodo. GP Curve Fitting también cuenta con este método utilizando la misma probabilidad que el método anterior, sólo que sobre un individuo en lugar de hacerlo sobre sus nodos.

Hemos mencionado que GP Curve Fitting cuenta con los dos métodos de mutación, mutación por nodo y mutación por subárbol. Ahora toca explicar cómo hicimos que interactuaran ambos. Como ya se ha explicado antes, la mutación es una forma de explotación del espacio de búsqueda, puesto que realiza pequeños cambios en los individuos; sin embargo, si lo pensamos un poco, la mutación por subárbol puede llegar a cambiar una parte significativa del individuo. Tampoco podemos decir que este método de mutación no sea útil, puesto que ayuda a generar diversidad en la población, lo cual es muy importante en la programación genética, por lo que no podemos dejar fuera ninguno de los dos métodos.

GP Curve Fitting incorpora los dos métodos asignándoles una probabilidad de ocurrencia. La probabilidad más alta (0.95) es para la mutación por nodo mientras que la mutación por sub-árbol tiene poca probabilidad de ocurrir (0.05). Así podemos aprovechar las cualidades de ambos métodos.

4.7. Criterio de término de ejecución

GP Curve Fitting cuenta con una variable global llamada *valor-finalizar* con la cual podemos establecer el valor mínimo en el que se debe detener el programa (condición de paro). El valor establecido en la variable global se compara con la aptitud del mejor individuo de la población. En cuanto este individuo tenga un valor menor o igual al límite establecido, la ejecución finaliza.

El programa ejecuta el número de generaciones que se establecen en la variable global *numero-generaciones*. Una vez evolucionada la población hasta ese punto, el programa termina, aunque aún no se haya cumplido la condición de paro. Para establecer el número de generaciones y la condición de paro se requieren experimentos previos.

CAPÍTULO V

RESULTADOS OBTENIDOS

Contenido

La programación genética ha demostrado ser un algoritmo evolutivo capaz de resolver problemas en donde nos enfrentamos a espacios de búsqueda enormes y donde un método determinístico quedaría atrapado en máximos locales. En este capítulo mostraremos los resultados de los experimentos realizados durante nuestra investigación.

5.1. Estudios previos

En el estado del arte (Capítulo II) encontramos varios problemas de ajuste de curvas que se han resuelto mediante heurísticas. Mostramos la réplica de dos experimentos. El primero realizado por Kamal y Eassa (2002) y el segundo por Senturk (2009).

El experimento propuesto por Kamal y Eassa (2002) fue resuelto con un sistema de programación genética; se obtuvo un error cuadrático medio **de 0.003 en 500 generaciones**. Los datos usados para el experimento se muestran en la Tabla 5.1. El error cuadrático medio que se obtuvo al replicar el experimento (Kamal y Eassa, 2002) con el sistema GP Curve Fitting fue en promedio **de 0.002 y se obtuvo en 500 generaciones** en un total de 30 experimentos.

Tabla 5.1: Experimento de Kamal y Eassa (2002)

X	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Y	0.00	0.30	0.40	0.55	0.63	0.71	0.77	0.84	0.89	0.95	1.00

El segundo experimento fue realizado por Senturk (2009) en el cual se buscaron los coeficientes de la función $y = -0.0140x^3 + 1.1169x^2 - 2.2015x + 0.9596$. Senturk (2009) obtuvo un error cuadrático de 0.007 con un algoritmo genético. Replicamos el experimento y obtuvimos con GP Curve Fitting un error en promedio de 0.005 en 500 generaciones en un total de 30 experimentos.

Los trabajos realizados en el estado del arte son enfocados a conjuntos pequeños de datos. Debido a que en nuestro trabajo necesitamos analizar la función resultante para determinar qué variables son relevantes, entonces tenemos que considerar un conjunto mayor de puntos a ajustar. El sistema de programación genética implementado en esta investigación es una mejora a los sistemas existentes que sólo tienen como incógnita el valor de una variable. El sistema que creamos trabaja con múltiples variables independientes que pueden generar ajustes de curvas muy precisos a curvas complejas. Además de generar un análisis de la solución encontrada para identificar la relevancia de cada variable involucrada en la solución.

5.2. Sobre los datos usados para los experimentos

Para los experimentos que mostraremos utilizamos series de datos de tipo económico, mismos que tomamos de la página www.indexmundi.com. Estos datos están divididos en 10 categorías: energía, bebidas, cereales, frutas, carnes, pescados y mariscos, azúcar, aceites vegetales y harinas de proteína, materias primas agrícolas, y metales. El detalle de cada categoría se presenta en la Tabla 5.2 y en la Tabla 5.3.

Tabla 5.2: Mercancías agrupadas por categoría (a)

Categoría	Mercancía	Variable usada
Aceites y harinas	Aceite de coco	X_1
	Aceite de colza	X_2
	Aceite de girasol	X_3
	Aceite de maní	X_4
	Aceite de oliva	X_5
	Aceite de palma	X_6
	Aceite de palmiste	X_7
	Aceite de soja	X_8
	Harina de pescado	X_9
	Harina de soja	X_{10}
	Maní	X_{11}
	Soja	X_{12}
Azúcar	Azúcar	X_{13}
	Azúcar precio de importación americano	X_{14}
	Azúcar precio de importación europeo	X_{15}
Bebidas	Café arábiga suave	X_{16}
	Café robusta	X_{17}
	Grano de cacao	X_{18}
	Té	X_{19}
Carnes	Carne	X_{20}
	Aves de corral	X_{21}
	Cerdo	X_{22}
	Cordero	X_{23}
Cereales	Arroz	X_{24}
	Cebada	X_{25}
	Maíz	X_{26}
	Sorgo	X_{27}
	Trigo	X_{28}
	Trigo soft red winter	X_{29}

Tabla 5.3: Mercancías agrupadas por categoría (b)

Categoría	Mercancía	Variable usada
Energía	Carbón australiano térmico	X_{30}
	Carbón colombiano	X_{31}
	Carbón sudafricano	X_{32}
	Diésel	X_{33}
	Fueloil	X_{34}
	Gas natural	X_{35}
	Gas natural licuado de indonesia	X_{36}
	Gas natural ruso	X_{37}
	Gasolina	X_{38}
	Gasolina de aviación	X_{39}
	Gasolina reformulada	X_{40}
	Petróleo crudo	X_{41}
	Petróleo crudo brent	X_{42}
	Petróleo crudo de Dubai Fateh	X_{43}
	Petróleo curdo del oeste de Texas	X_{44}
	Propano	X_{45}
Fruta	Naranja	X_{46}
	Plátano	X_{47}
Pescados y mariscos	Camarón	X_{48}
	Salmón	X_{49}
Materias primas agrícolas	Algodón	X_{50}
	Copra	X_{51}
	Cuero	X_{52}
	Goma	X_{53}
	Lana fina	X_{54}
	lana gruesa	X_{55}
	Madera contrachapada	X_{56}
	Madera dura	X_{57}
	Madera dura aserrada	X_{58}
	Madera suave	X_{59}
	Madera suave aserrada	X_{60}
	Pulpa de celulosa	X_{61}
Metales	Aluminio	X_{62}
	Cobre	X_{63}
	Estaño	X_{64}
	Mineral de hierro	X_{65}
	Níquel	X_{66}
	Oro	X_{67}
	Plata	X_{68}
	Plomo	X_{69}
	Uranio	X_{70}
	Zinc	X_{71}

Cada serie de datos contiene los datos del precio mensual del producto desde noviembre del 2008 hasta noviembre del 2012 en moneda nacional, reuniendo un total de 49 datos por serie. Se recabaron un total de 71 series de datos de las diferentes categorías.

Las series de datos que se obtuvieron de cada categoría serán parte del conjunto de terminales del sistema de programación genética, es decir, serán las variables independientes que pudieran quedar en la solución generada por el sistema.

Por otra parte, necesitamos los datos que queremos ajustar. Los índices económicos o indicadores económicos son una medida estadística creada para exponer los cambios de una variable económica o un grupo de variables relacionadas en un tiempo determinado y a través de esto explicar una situación económica. El uso de índices genera algunas dificultades. Cuando se sintetiza una realidad económica mediante números es inevitable que se omitan elementos; es así como los índices darán una imagen cercana a la realidad sin llegar a ser igual a ella.

Ahora bien, si un índice se genera a partir de un grupo de variables relacionadas, entonces es posible generar una función matemática (con programación genética) que encuentre esas relaciones entre variables para poder ajustar los datos de un índice cualquiera. Por esta razón, usaremos la categoría de índices compuesta por: índice de precios de materias primas, índice de precios de alimentos, índice de precios de bebidas, índice de precios de combustibles, índice de insumos industriales. Cada una de estas series de datos pertenece a alguna categoría de productos ya mencionadas y cada índice se utilizó para un experimento. Esto es, el índice (variable dependiente) es la curva que queremos ajustar y las series de datos de los productos se usarán como variables independientes para formar una expresión matemática que resuelva de la mejor manera posible nuestro ajuste.

Para evaluar los experimentos utilizaremos el coeficiente de determinación y la fórmula de error cuadrático medio. En todos los experimentos se usaron los siguientes parámetros: población de 500 individuos, un total de 5000 generaciones, probabilidad de recombinación de 0.9, probabilidad de mutación de 0.01 y una profundidad de árbol inicial máxima de 6. Para delimitar el crecimiento de los árboles se utilizó el número de

nodos, permitiendo a cada árbol tener un máximo de 500. Si un árbol sobrepasa el límite, es penalizado, con lo cual se afecta la aptitud del mismo convirtiéndolo en una peor solución.

Cada experimento fue repetido 30 ocasiones. En cada experimento mostramos uno de los experimentos que obtuvo un error cuadrático medio igual o aproximado al promedio.

Como aportación extra a la investigación, se presenta un primer enfoque al análisis de las funciones matemáticas creadas por el sistema de PG. Se muestra en cada experimento una tabla en donde se encuentran las principales variables independientes (materias primas) que tienen mayor relevancia para el ajuste. La forma en que proponemos obtener la relevancia de cada variable se explica más adelante. Un análisis más elaborado de las funciones está considerado como trabajo futuro, donde se podría profundizar más en analizar minuciosamente las propiedades de la función creada por el sistema de PG.

5.3. Experimento 1. Ajuste del índice de precios de materias primas agrícolas

En el primer experimento ajustaremos el índice de precios de materias primas agrícolas.

El error cuadrático medio obtenido fue de 0.3 y el coeficiente de 0.9998. Los datos del ajuste se muestran en la Tabla 5.4. Como podemos observar en la tabla los puntos generados por la función (f(x)) son muy cercanos a los puntos de los datos experimentales (*y*) lo que podemos ver gráficamente en la Figura 5.1.

Después de obtener la función matemática, necesitamos saber qué variables independientes son las que tienen mayor relevancia para el ajuste realizado; por tanto, ideamos un método mediante el cual obtenemos un valor numérico para lograr el objetivo. El método consiste en "neutralizar" una variable independiente (cambiando los datos de la serie por el promedio de la misma) y recalculamos la aptitud de la solución. Una vez obtenido el nuevo error se lo restamos al error original (de la función sin variables neutralizadas) y eso lo dividimos entre el error total acumulado de todas la

variables neutralizadas. Por último consideramos necesario no omitir el valor de correlación: entonces, obtenemos el producto del resultado hasta el momento y la correlación entre la variable dependiente e independiente. Este proceso se realiza con cada una de las variables independientes.

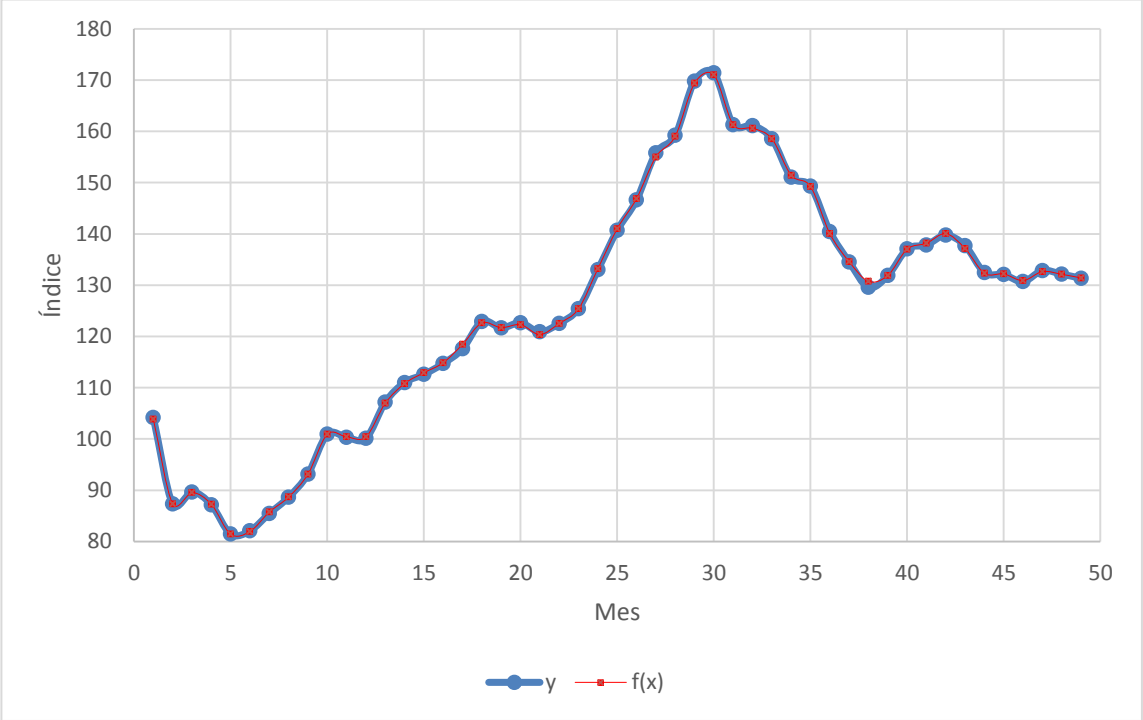

Figura 5.1: Gráfica del ajuste del índice de precios de materias primas agrícolas

En el análisis de la función que se muestra en la Figura 5.1, la variable x7 que se refiere al precio del aceite de palmiste de la categoría aceites y harinas obtuvo un porcentaje de importancia del 79.13 %. Esto quiere decir, que al anular la variable del precio del aceite de palmiste el ajuste se ve deteriorado un 79.13% lo cual indica que esta variable es muy importante en la función. Otros porcentajes menores son los de la variable x62 que corresponde al precio del aluminio de la categoría de metales con un 13.17% de aportación, la variable x33 que corresponde al diésel, de la categoría de energía, con un porcentaje de 2.7% de aportación y la variable x52 que se refiere al precio del cuero, de la categoría de materias primas agrícolas, con un 1.6 % de aportación. Podemos ver los resultados en la Tabla 5.5 junto con el coeficiente de correlación correspondiente con el

ajuste. Notemos que el porcentaje de importancia no necesariamente es equivalente al coeficiente de correlación.

Tabla 5.4: Datos obtenidos del ajuste del índice de precios de materias primas agrícolas, donde "y" representa los datos experimentales y "f(x)" la función generada

No. dato	Y	f(x)
1	104.19	103.85
2	87.32	87.35
3	89.66	89.54
4	87.17	87.26
5	81.45	81.44
6	82.10	81.96
7	85.45	85.76
8	88.66	88.68
9	93.14	93.15
10	100.91	100.92
11	100.32	100.41
12	100.11	100.41
13	107.18	106.98
14	111.01	110.81
15	112.60	112.93
16	114.72	114.88
17	117.59	118.47
18	122.91	122.63
19	121.68	121.71
20	122.64	122.29
21	120.88	120.38
22	122.54	122.53
23	125.43	125.41
24	132.99	133.22

No. dato	y	f(x)
25	140.70	141.01
26	146.61	146.90
27	155.78	155.01
28	159.20	159.03
29	169.80	169.38
30	171.37	171.03
31	161.31	161.30
32	161.12	160.62
33	158.56	158.56
34	151.03	151.43
35	149.32	149.25
36	140.45	140.06
37	134.54	134.58
38	129.57	130.74
39	131.86	131.90
40	137.05	137.02
41	137.83	138.17
42	139.74	140.05
43	137.70	137.14
44	132.45	132.31
45	132.06	132.19
46	130.70	130.90
47	132.84	132.65
48	132.15	132.12
49	131.33	131.40

Tabla 5.5: Análisis de la función obtenida en el ajuste del índice de precios de materias primas agrícolas

Materia prima	Categoría	Relevancia	Correlación
Aceite de palmiste	Aceites y harinas	79.13%	0.87
Aluminio	Metales	13.17%	0.88
Diésel	Energía	2.70%	0.77
Cueros	Materias primas agrícolas	1.60%	0.82

Podemos observar que el índice que ajustamos corresponde a la categoría de materias primas agrícolas y los resultados apuntan mayor importancia a otras categorías. Esto no es malo, simplemente recordemos que la PG se caracteriza por encontrar formas de resolver problemas que el humano difícilmente pensaría.

Otro punto que debemos destacar es que nada nos garantiza que entre un ajuste y otro generado por la programación genética obtengamos el mismo porcentaje de aportación de las variables.

Se realizaron 30 ajustes sobre el mismo experimento, de los cuales se obtuvo un error promedio de 0.3.

5.4. Experimento 2. Ajuste del índice de precios de alimentos

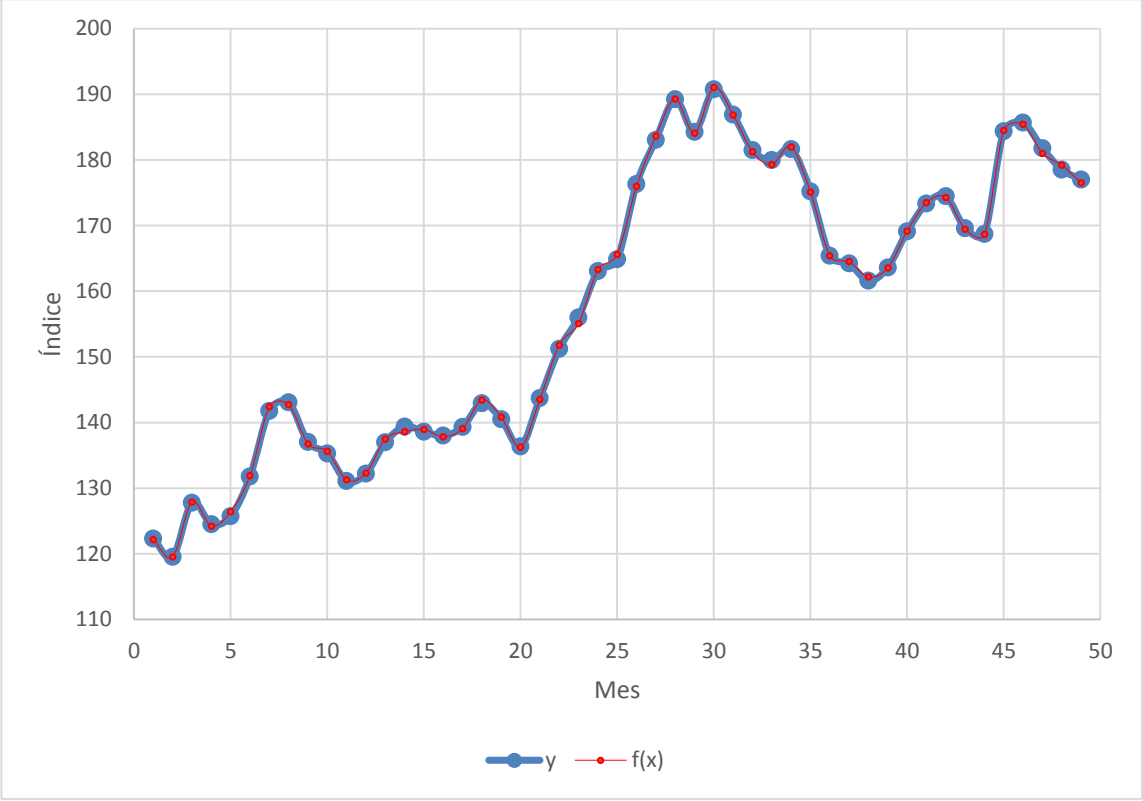

Figura 5.2: Gráfica del ajuste del índice de precios de alimentos

El segundo experimento es el ajuste del índice de precios de alimentos. Este índice es un número que podría verse como el resumen de todos los precios relacionados con los alimentos representado por un único valor.

El valor de error cuadrático medio obtenido fue de 0.4 y el coeficiente de determinación fue de 0.9996. Los datos del índice de precios de alimentos y los de la función de ajuste se muestran en la Tabla 5.6. También podemos ver la gráfica del ajuste en la Figura 5.2. Observemos que los datos que resultan del ajuste son muy cercanos a la curva original.

El resultado del análisis de la función se muestra en la Tabla 5.7. En un principio se trató de despreciar las variables que tuvieran poca correlación con los datos a ajustar.

Tabla 5.6: Datos obtenidos del ajuste del índice de precios de alimentos, donde "y" representa los datos experimentales y "f(x)" la función generada

Mes	y	f(x)
1	122.30	122.15
2	119.51	119.53
3	127.79	127.87
4	124.51	124.22
5	125.74	126.41
6	131.76	131.91
7	141.75	142.38
8	143.08	142.72
9	137.02	136.76
10	135.29	135.62
11	131.08	131.26
12	132.21	132.29
13	136.97	137.46
14	139.39	138.60
15	138.60	138.93
16	138.00	137.83
17	139.31	139.07
18	142.94	143.43
19	140.51	140.84
20	136.34	136.25
21	143.71	143.53
22	151.21	151.81
23	156.00	155.09
24	163.04	163.28

Mes	y	f(x)
25	164.85	165.61
26	176.33	175.99
27	183.07	183.64
28	189.22	189.27
29	184.24	184.05
30	190.77	191.05
31	186.91	186.83
32	181.49	181.23
33	179.97	179.30
34	181.64	181.99
35	175.20	175.08
36	165.42	165.42
37	164.23	164.50
38	161.58	162.16
39	163.62	163.56
40	169.11	169.14
41	173.34	173.48
42	174.47	174.27
43	169.60	169.44
44	168.73	168.64
45	184.37	184.51
46	185.68	185.43
47	181.78	181.00
48	178.53	179.22
49	177.02	176.51

Tabla 5.7: Análisis de la función obtenida en el ajuste del índice de precios de alimentos

Materia prima	Categoría	Relevancia	Correlación
Trigo	Cereales	10.35%	0.8
Fueloil	Energía	10.35%	0.84
Gas natural licuado de indonesia	Energía	10.35%	0.71
Aceite de palmiste	Aceites y harinas	8.81%	0.79
Carne	Carnes	8.79%	0.82
Petróleo crudo del oeste de Texas	Energía	8.32%	0.78
Zinc	Metales	7.05%	0.53
Petróleo crudo Brent	Energía	4.03%	0.86
Harina de pescado	Aceites y harinas	2.91%	0.32
Uranio	Metales	2.70%	0.38
Naranja	Fruta	1.86%	0.01
Aceite de coco	Aceites y harinas	1.11%	0.79
Azúcar	Azúcar	1.11%	0.64

Sin embargo, en los experimentos realizados se observa cómo la correlación no necesariamente indica qué tan relevante es una variable dentro de una función matemática generada en un ajuste de curvas. Esto se debe a que el sistema de PG puede generar ciertas relaciones entre las variables, haciéndolas más o menos relevantes para la función matemática que se ha generado. Por lo tanto, dejamos de asumir que la correlación tenga una relación directa con la relevancia de cada variable independiente.

Se realizaron 30 ejecuciones del mismo experimento de los cuales se obtuvo un error cuadrático medio promedio de 0.4 y un coeficiente de determinación promedio de 0.9995.

5.5. Experimento 3 - Ajuste del índice de precios de bebidas

Para el tercer experimento se utilizaron los datos del índice de precios de bebidas. Tomemos en cuenta que un índice no es una medida en particular (pesos, kilos, litros), sino simplemente un valor numérico que nos permite medir y analizar un fenómeno económico.

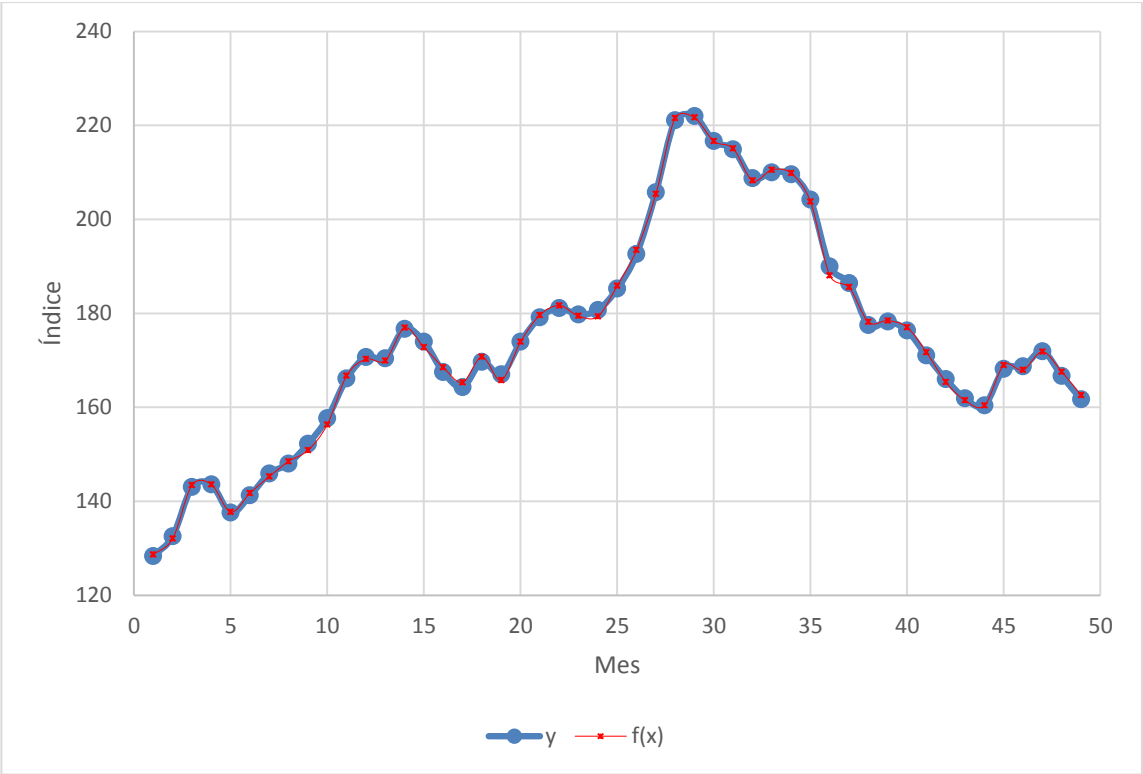

Figura 5.3: Gráfica del ajuste del índice de precios de bebidas

El error cuadrático medio obtenido en el experimento que mostraremos a continuación es de 0.7 y el coeficiente de determinación que se obtuvo fue de 0.9989. Podemos verificar los datos del ajuste en la Tabla 5.9. La gráfica de la función generada y del índice que se ajustó la observamos en la Figura 5.3.

El análisis de la función matemática resultante del ajuste de curvas se muestra en la Tabla 5.8. Notemos que la gasolina es una variable, que en esta función generada, resultó ser muy relevante.

Tabla 5.8: Análisis de la función obtenida en el ajuste del índice de precios de bebidas

Materia prima	Categoría	Importancia	Correlación
Gasolina	Energía	55.06%	0.54
Azúcar	Azúcar	19.17%	0.73
Café arábiga suave	Bebidas	15.06%	0.88
Aves de corral	Carnes	4.95%	0.01

Tabla 5.9: Datos obtenidos del ajuste del índice de precios de bebidas, donde "y" representa los datos experimentales y "f(x)" la función generada

Mes	y	f(x)
1	128.33	128.66
2	132.54	132.03
3	143.01	143.44
4	143.59	143.60
5	137.57	137.70
6	141.23	141.72
7	145.87	145.26
8	148.02	148.49
9	152.22	150.84
10	157.65	156.34
11	166.09	166.71
12	170.70	170.25
13	170.43	169.96
14	176.70	176.92
15	173.97	172.77
16	167.48	168.51
17	164.30	165.31
18	169.63	170.73
19	167.00	165.76
20	173.92	173.95
21	179.12	179.63
22	181.08	181.67
23	179.75	179.43
24	180.71	179.35

Mes	y	f(x)
25	185.24	185.86
26	192.58	193.46
27	205.72	205.41
28	221.05	221.57
29	221.99	221.66
30	216.63	216.64
31	214.91	215.06
32	208.74	208.26
33	209.96	210.55
34	209.55	209.82
35	204.16	203.75
36	189.95	188.02
37	186.39	185.56
38	177.49	178.14
39	178.25	178.44
40	176.30	177.03
41	171.01	171.70
42	165.98	165.37
43	161.87	161.45
44	160.38	160.39
45	168.18	168.97
46	168.69	167.92
47	171.90	171.84
48	166.67	167.57
49	161.71	162.60

Se ejecutó el mismo experimento en 30 ocasiones con un error cuadrático medio promedio de 0.7 y un coeficiente de determinación promedio de 0.9986.

5.6. Experimento 4 - Ajuste del índice de precios de combustibles

En este experimento realizaremos el ajuste de curvas para el índice de precios de combustibles.

Recordemos que cada en cada ajuste 71 variables independientes de las 10 categorías estarán disponibles para que el sistema de programación genética genere la función. Por lo tanto, la función puede contener variables de otras categorías, no necesariamente de

una sola. Una vez generada la función el sistema puede analizarla y regresarnos un índice que indica qué peso tiene cada variable independiente dentro de la expresión.

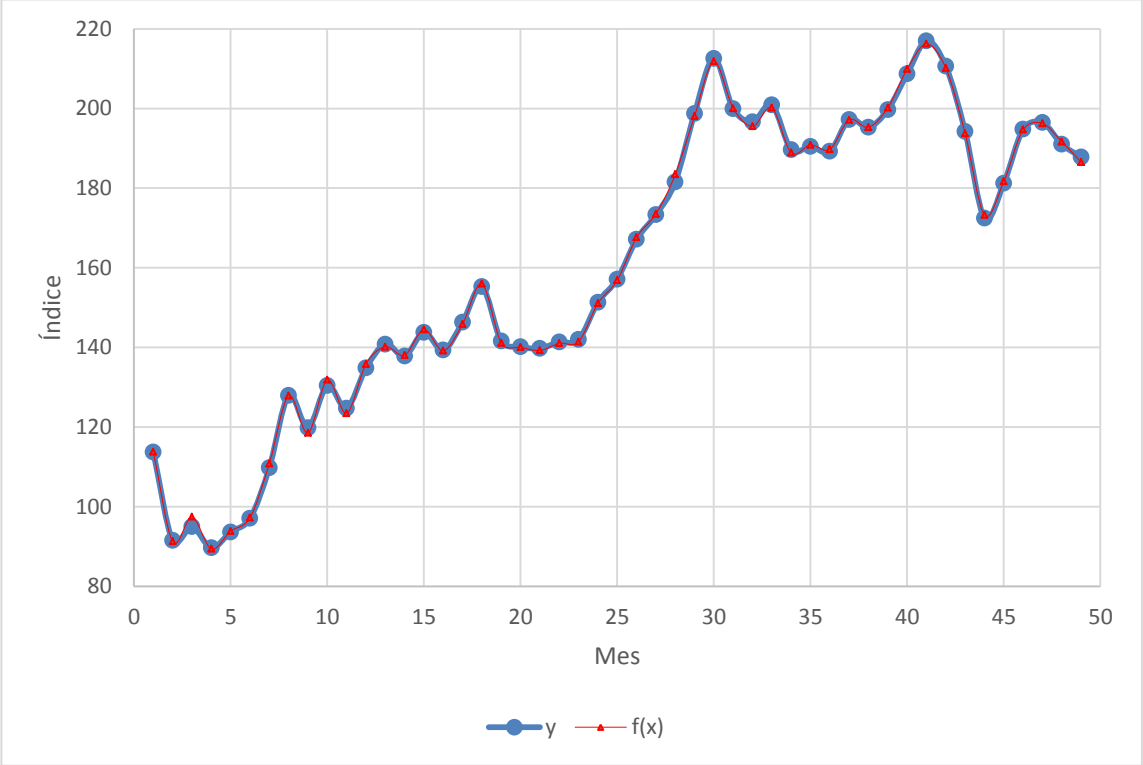

Figura 5.4: Gráfica del ajuste del índice de precios de combustibles

El error cuadrático medio que se obtuvo en este experimento fue de 0.8 y un coeficiente de determinación de 0.9995. Los datos del índice de precios de combustibles y de la función de ajuste están en la Tabla 5.11 y las gráficas correspondientes se encuentran en la Figura 5.4.

Tabla 5.10: Análisis de la función obtenida en el ajuste del índice de precios de combustibles

Materia prima	Categoría	Importancia	Correlación
Gasolina	Energía	29.98%	0.95
Cobre	Metales	29.91%	0.89
Lana fina	Materias primas agrícolas	19.39%	0.93
Arroz	Cereales	8.75%	0.01

El análisis de la función se puede ver en la Tabla 5.10. El experimento se repitió 30 ocasiones con un error cuadrático medio promedio de 0.8 y un coeficiente de determinación de 0.9994.

Tabla 5.11: Datos obtenidos del ajuste del índice de precios de combustibles, donde "y" representa los datos experimentales y "f(x)" la función generada

Mes	y	f(x)
1	113.70	113.79
2	91.55	91.26
3	94.99	97.44
4	89.70	89.40
5	93.64	93.86
6	97.05	97.21
7	109.78	110.83
8	127.94	127.84
9	119.83	118.51
10	130.44	131.83
11	124.74	123.44
12	134.81	135.81
13	140.82	140.17
14	137.84	138.04
15	143.74	144.49
16	139.30	139.16
17	146.29	145.88
18	155.22	155.94
19	141.62	141.10
20	140.14	140.04
21	139.75	139.23
22	141.35	141.04
23	141.95	141.46
24	151.33	151.05

Mes	y	f(x)
25	157.12	156.86
26	167.13	167.62
27	173.30	173.50
28	181.53	183.42
29	198.75	198.09
30	212.57	211.81
31	199.95	200.02
32	196.65	195.51
33	200.92	200.13
34	189.65	188.91
35	190.45	190.86
36	189.22	189.73
37	197.22	197.13
38	195.28	195.28
39	199.69	200.18
40	208.65	209.88
41	216.98	216.22
42	210.62	210.15
43	194.23	193.72
44	172.47	173.23
45	181.19	181.65
46	194.79	194.67
47	196.48	196.26
48	191.02	191.67
49	187.80	186.52

5.7. Experimento 5 - Ajuste del índice de precios de insumos industriales

Ahora ajustaremos el índice de precios de insumos industriales. El experimento terminó con un error cuadrático medio de 1.3 y un coeficiente de determinación de 0.9983. También podemos ver los datos del índice y de la función de ajuste en forma gráfica en la Figura 5.5.

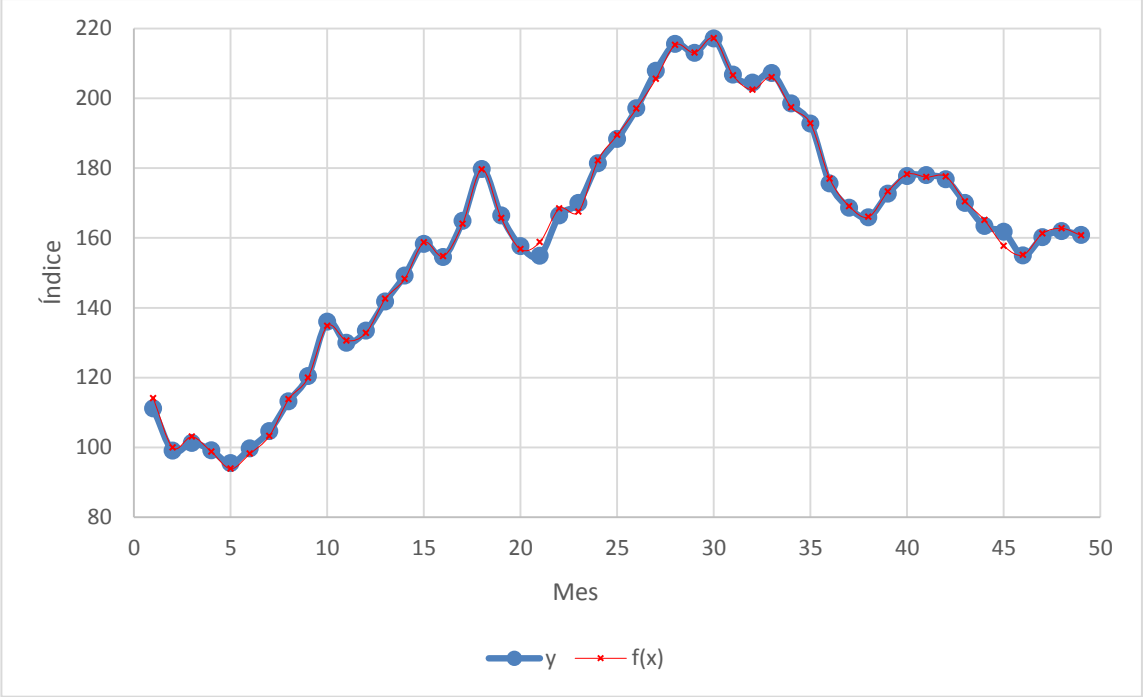

Figura 5.5: Gráfica del ajuste del índice de precios de insumos industriales

El análisis de la función matemática que corresponde a este experimento lo podemos ver en la Tabla 5.12.

Tabla 5.12: Análisis de la función obtenida en el ajuste del índice de precios de insumos industriales

Materia prima	Categoría	Importancia	Correlación
Aluminio	Metales	23.52%	0.94
Lana gruesa	Materias primas agrícolas	21.39%	0.73
Pulpa de celulosa	Materias primas agrícolas	20.80%	0.8
Harina de pescado	Aceites y harinas	15.65%	0.45
Maíz	Cereales	15.32%	0.56
Gas natural ruso	Energía	3.58%	0.01

El experimento se ejecutó 30 veces con un error cuadrático medio promedio de 1.3 y un coeficiente de determinación promedio de 0.9984. Podemos ver los resultados del ajuste en la Tabla 5.13.

Tabla 5.13: Datos obtenidos del ajuste del índice de precios de insumos industriales, donde "y" representa los datos experimentales y "f(x)" la función generada

Mes	y	f(x)
1	111.13	114.04
2	99.04	99.97
3	101.27	103.06
4	99.18	98.79
5	95.43	93.89
6	99.67	98.20
7	104.67	103.24
8	113.18	113.77
9	120.36	119.92
10	135.98	134.77
11	129.90	130.60
12	133.42	132.79
13	141.71	142.53
14	149.20	148.22
15	158.23	158.70
16	154.50	154.76
17	164.85	164.01
18	179.73	179.58
19	166.36	165.67
20	157.61	156.76
21	154.84	158.82
22	166.37	168.41
23	170.02	167.48
24	181.35	182.18

Mes	y	f(x)
25	188.28	189.46
26	197.11	197.05
27	207.86	205.52
28	215.54	215.27
29	213.00	213.04
30	217.07	217.19
31	206.69	206.53
32	204.43	202.44
33	207.14	206.02
34	198.52	197.41
35	192.74	192.78
36	175.55	176.91
37	168.64	169.10
38	165.88	166.00
39	172.61	173.26
40	177.72	178.24
41	177.95	177.41
42	176.75	177.59
43	170.00	170.47
44	163.36	165.12
45	161.72	157.71
46	154.97	155.14
47	160.17	161.24
48	161.91	162.75
49	160.82	160.71

5.8. Experimento 6 – Ajuste del índice de precios de alimentos y bebidas

En el siguiente experimento se ajustan los datos del índice de precios de alimentos y bebidas. Como el nombre lo indica, está formado de las relaciones tanto de la categoría de alimentos como de la categoría de bebidas.

Como ya se explicó en experimentos anteriores, el sistema de PG puede utilizar las variables de todas las categorías y no necesariamente usará las variables que corresponden al índice que se está ajustando.

Tabla 5.14: Datos obtenidos del ajuste del índice de precios de alimentos y bebidas, donde "y" representa los datos experimentales y "f(x)" la función generada

Mes	y	f(x)
1	122.89	122.68
2	120.78	120.74
3	129.28	130.11
4	126.38	126.24
5	126.9	127.20
6	132.69	132.78
7	142.16	142.60
8	143.56	143.30
9	138.51	138.30
10	137.48	136.84
11	134.5	134.84
12	135.97	136.27
13	140.24	140.17
14	143.03	142.49
15	142.05	142.03
16	140.88	140.77
17	141.76	142.24
18	145.55	146.08
19	143.1	143.33
20	140.01	140.04
21	147.17	146.84
22	154.13	154.81
23	158.32	158.02
24	164.77	164.80

Mes	y	f(x)
25	166.85	166.79
26	177.92	177.71
27	185.29	185.33
28	192.33	192.38
29	187.93	187.35
30	193.3	193.00
31	189.65	190.44
32	184.16	184.83
33	182.9	182.98
34	184.37	183.64
35	178.03	178.91
36	167.82	167.84
37	166.39	166.17
38	163.13	163.06
39	165.05	165.27
40	169.81	169.84
41	173.12	172.58
42	173.64	173.47
43	168.84	168.21
44	167.91	168.57
45	182.79	182.91
46	184.02	183.87
47	180.81	181.08
48	177.37	176.99
49	175.52	175.74

El error cuadrático medio obtenido en el ajuste fue de 0.4, y el coeficiente de determinación fue de 0.9996. Los resultados del ajuste los podemos ver en la Tabla 5.14. La gráfica del ajuste se muestra en la Figura 5.6.

El análisis de la función matemática se presenta en la Tabla 5.15.

Tabla 5.15: Análisis de la función obtenida en el ajuste del índice de precios de alimentos y bebidas

Materia prima	Categoría	Importancia	Correlación
Plata	Metales	92.97%	0.91
Aceite de palmiste	Aceites y harinas	0.70%	0.81
Madera dura	Materias primas agrícolas	0.60%	0.47
Grano de cacao	Bebidas	0.50%	0.01

El experimento fue repetido 30 veces con un error cuadrático medio promedio de 0.4 y un coeficiente de determinación promedio de 0.9995.

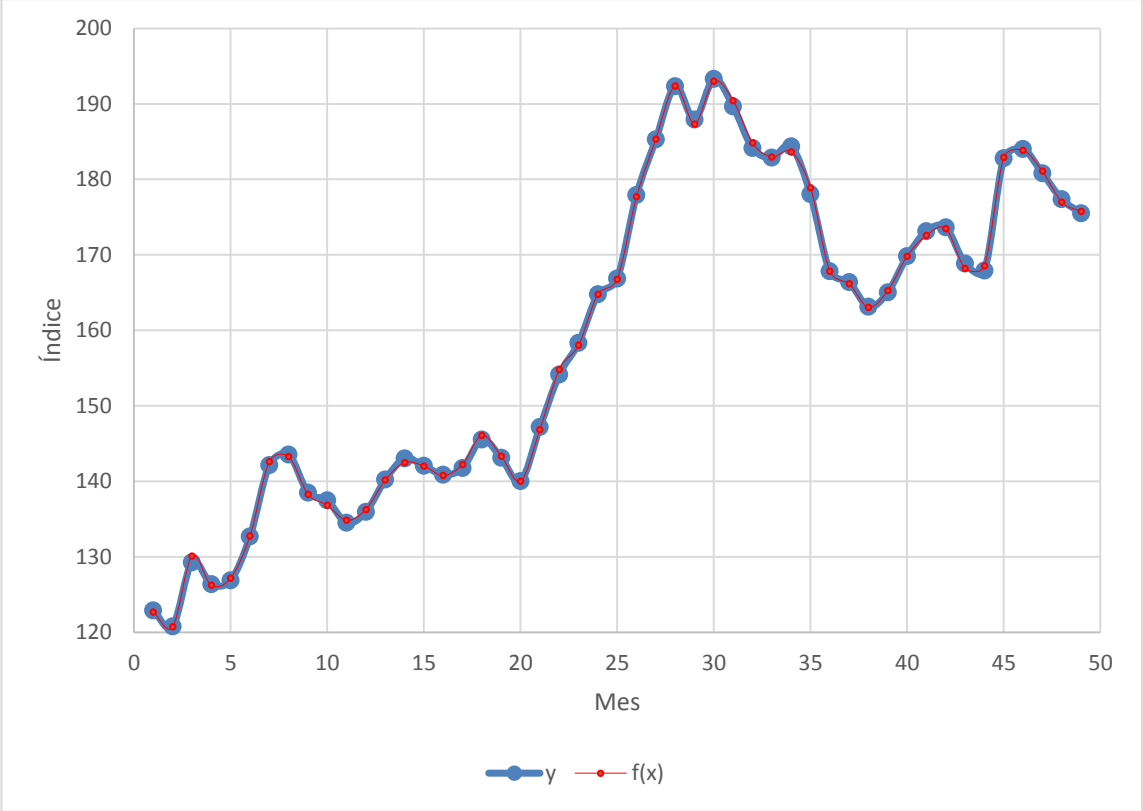

Figura 5.6: Gráfica del ajuste del índice de precios de alimentos y bebidas

5.9. Experimento 7 – Ajuste del índice de precios de petróleo crudo

El experimento realiza el ajuste del índice de precios de petróleo crudo. Se puede notar que la gráfica del índice de precios de energía es muy parecida a la de este experimento. Sin embargo, si comparamos los datos (de la columna "y") de la Tabla 5.16 y la Tabla 5.11, veremos que varios datos ya no resultan ser tan parecidos. Por ello, no se puede utilizar la misma función que se creó en el experimento 4 a menos que al usuario no le importe que el ajuste se deteriore.

El error cuadrático medio que se obtuvo en este experimento fue de 0.4 y el coeficiente de determinación fue de 0.9999.

Tabla 5.16: Datos obtenidos del ajuste del índice de precios de petróleo crudo, donde "y" representa los datos experimentales y "f(x)" la función generada.

Mes	y	f(x)
1	101.24	101.28
2	77.71	77.60
3	82.58	82.74
4	78.83	79.09
5	87.89	88.15
6	94.55	94.77
7	109.28	108.96
8	129.99	130.54
9	121.64	121.83
10	134.68	135.08
11	128.47	128.15
12	139.21	138.97
13	145.82	146.40
14	140.86	140.80
15	144.95	143.78
16	140.40	140.17
17	148.94	149.12
18	158.13	158.34
19	142.15	142.53
20	140.45	140.49
21	139.96	140.25
22	142.57	141.79
23	143.08	143.00
24	153.57	153.27

Mes	y	f(x)
25	158.91	158.34
26	169.33	169.00
27	174.28	174.25
28	184.10	184.07
29	204.42	204.52
30	218.82	218.66
31	203.62	203.75
32	199.35	199.42
33	203.23	202.41
34	189.50	190.19
35	190.27	190.40
36	188.44	188.02
37	198.51	198.03
38	196.31	196.99
39	201.32	202.12
40	212.38	212.39
41	222.03	222.31
42	214.36	214.25
43	196.28	196.65
44	171.02	171.19
45	182.28	182.36
46	198.42	197.91
47	200.48	200.44
48	195.06	195.05
49	190.93	191.36

Los datos experimentales y los resultados de la función evaluada se muestran en la Tabla 5.16. La gráfica del ajuste se muestra en la Figura 5.7.

Tabla 5.17: Análisis de la función obtenida en el ajuste del índice de precios de petróleo crudo

Materia prima	Categoría	Importancia	Correlación
Plata	Metales	92.97%	0.91
Aceite de palmiste	Aceites y harinas	0.70%	0.81
Madera dura	Materias primas agrícolas	0.60%	0.47
Grano de cacao	Bebidas	0.50%	0.01

El análisis de la función matemática se muestra en la Tabla 5.17. Podemos observar que podemos ajustar el índice de precio del petróleo con variables que no están en la misma categoría.

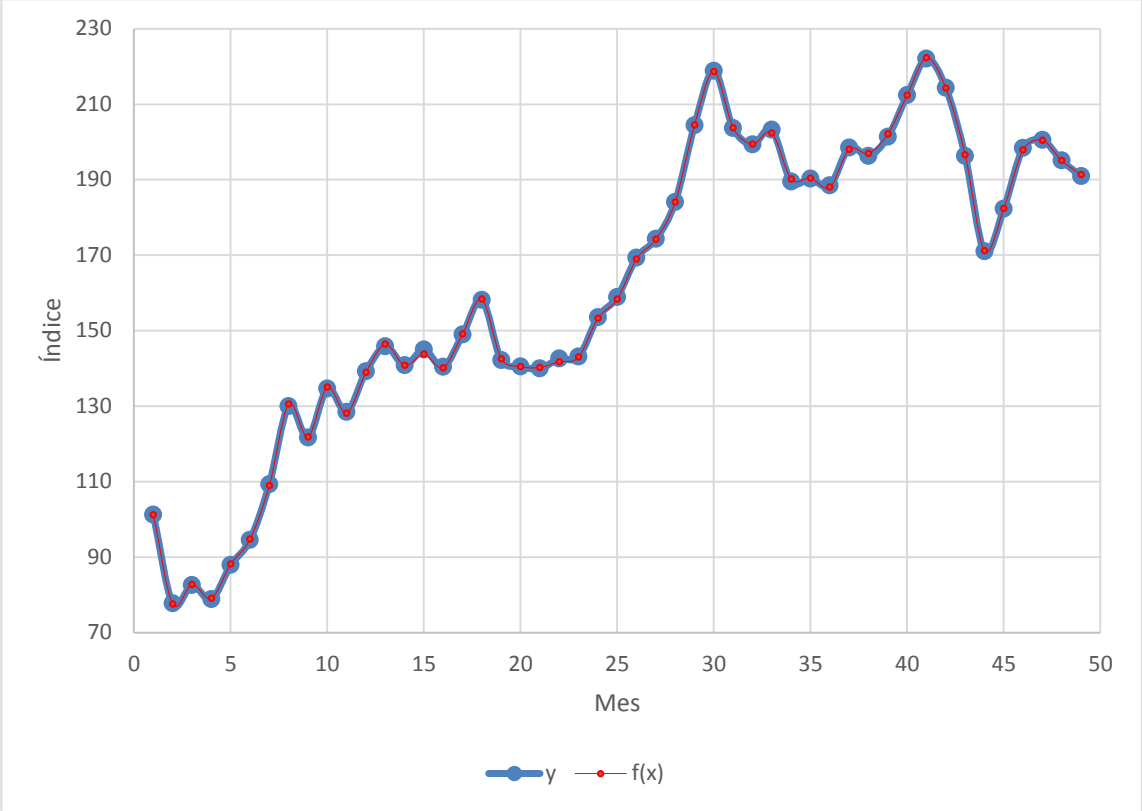

Figura 5.7: Gráfica del ajuste del índice de precios del petróleo crudo

5.10. Experimento 8 – Ajuste del índice de precios de no combustibles

En este experimento, tratamos de encontrar una función que aproxime lo más posible a los datos del índice de precios de no-combustibles. El índice de precios de no-combustibles está originalmente compuesto de las categorías alimentos, bebidas e insumos industriales.

En el momento en que el sistema de PG empiece a buscar soluciones, utilizará los datos de todas las categorías disponibles y regresará una solución. En experimentos pasados

hemos visto casos donde las variables que se consideran relevantes para la función matemática creada, no pertenecen a la categoría o categorías que abarca el indicador económico. Esto es debido a que la PG puede generar, en algunas ocasiones, soluciones hasta el momento desconocidas.

Tabla 5.18: Datos obtenidos del ajuste del índice de precios de no combustibles, donde "y" representa los datos experimentales y "f(x)" la función generada

Mes	y	f(x)
1	117.04	116.92
2	109.97	110.08
3	115.34	115.40
4	112.84	113.18
5	111.24	111.83
6	116.26	116.71
7	123.51	123.29
8	128.45	127.96
9	129.48	129.27
10	136.73	136.43
11	132.21	132.45
12	134.70	134.39
13	140.97	141.54
14	146.10	145.89
15	150.10	150.17
16	147.65	147.43
17	153.24	152.50
18	162.55	162.61
19	154.68	154.37
20	148.77	149.52
21	150.99	151.25
22	160.22	160.05
23	164.14	164.26
24	173.02	173.45

Mes	y	f(x)
25	177.51	177.80
26	187.47	187.65
27	196.52	195.84
28	203.88	204.11
29	200.41	199.73
30	205.12	204.57
31	198.12	198.00
32	194.24	194.64
33	194.96	195.49
34	191.41	191.79
35	185.35	185.05
36	171.67	171.37
37	167.51	167.87
38	164.50	165.22
39	168.81	168.31
40	173.75	174.05
41	175.52	175.92
42	175.19	175.57
43	169.42	169.02
44	165.65	165.26
45	172.31	171.49
46	169.57	169.17
47	170.54	170.58
48	169.68	169.61
49	168.21	168.42

El error cuadrático medio obtenido en este experimento fue de 0.4 y el coeficiente de determinación fue de 0.9997. En la Figura 5.8 se muestra la gráfica de la función encontrada y de los datos experimentales. El resultado de la función evaluada en los 49 puntos de datos se muestra en la Tabla 5.18, en comparación con los datos originales.

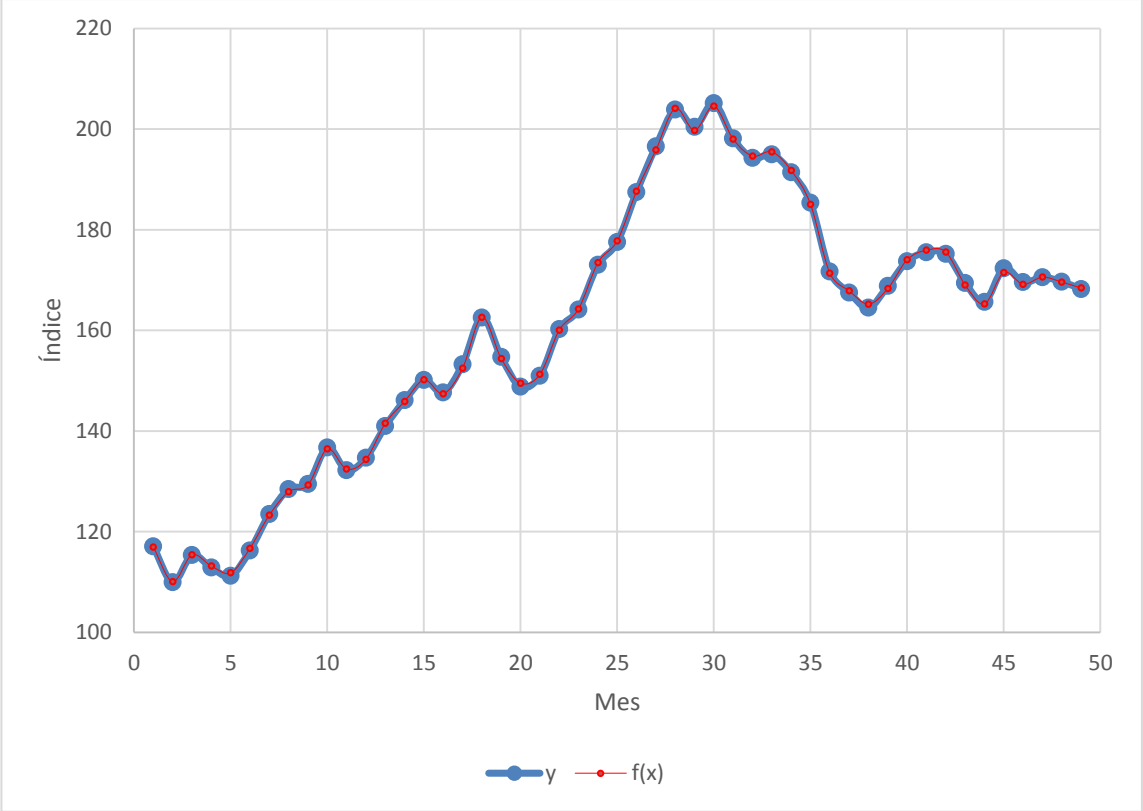

Figura 5.8: Gráfica del ajuste del índice de precios de no combustibles

El análisis de la función encontrada se presenta en la Tabla 5.19.

El experimento fue repetido 30 veces con un error cuadrático medio promedio de 0.4 y un coeficiente de determinación promedio de 0.9996.

Tabla 5.19: Análisis de la función obtenida en el ajuste del índice de precios de no combustibles

Materia prima	Categoría	Importancia	Correlación
Plata	Metales	82.99%	0.85
Aves de corral	Carnes	8.22%	0.01
Harina de pescado	Aceites y harinas	1.70%	0.42
Cerdo	Carnes	1.04%	0.66

5.11. Experimento 9 – Ajuste del índice de precios general de mercancías

Mostramos a continuación el ajuste del índice de precios general de mercancías, que incluye al índice de precios de combustibles y de no-combustibles.

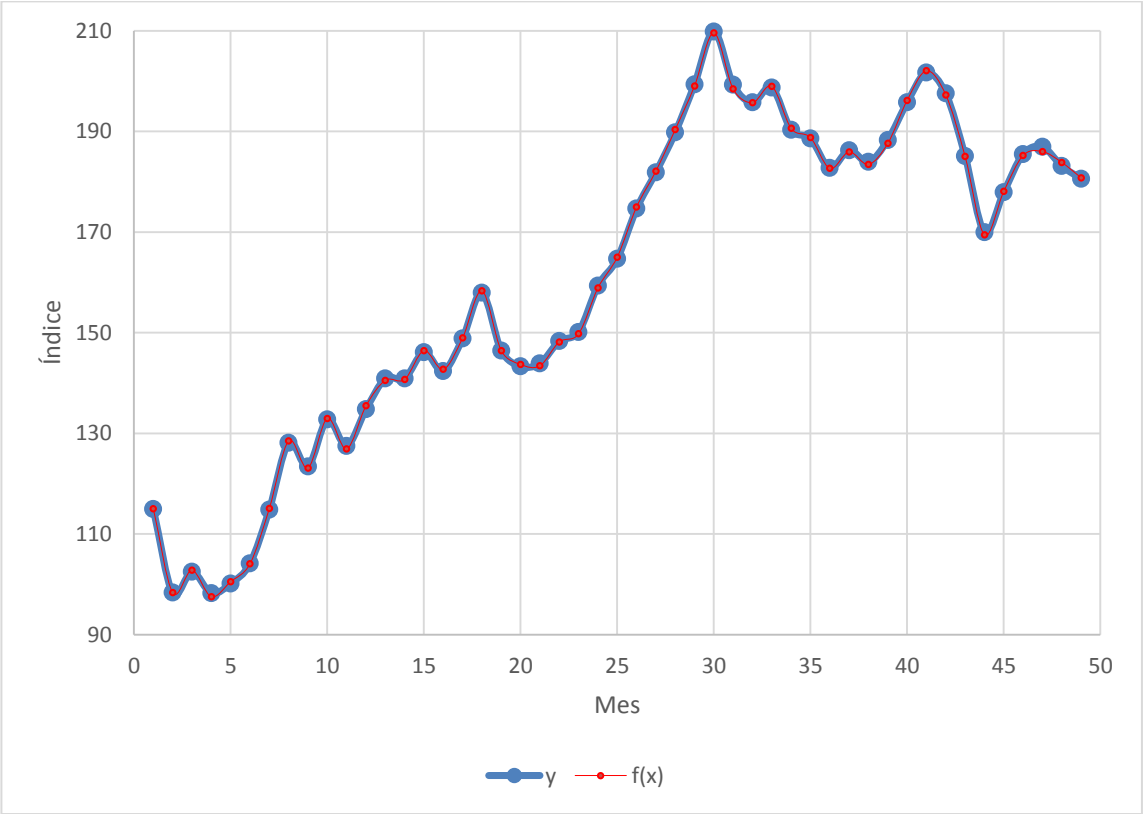

Figura 5.9: Gráfica del ajuste del índice de precios general de mercancías

El error cuadrático medio obtenido para este experimento fue de 0.4 y el coeficiente de determinación fue de 0.9999. En la Figura 5.9 mostramos la gráfica de los datos originales y del resultado de la función creada. Los datos de la evaluación de la función en comparación con los datos originales se muestran en la Tabla 5.20.

El análisis de la función se presenta en la Tabla 5.21. Notemos que esta vez la correlación de todas las materias primas que resultaron relevantes es considerablemente baja. Aun así es posible encontrar cierta relación entre variables para realizar un ajuste razonablemente bueno.

Tabla 5.20: Datos obtenidos del ajuste del índice de precios general de mercancías, donde "y" representa los datos experimentales y "f(x)" la función generada

Mes	y	f(x)
1	114.93	115.04
2	98.34	98.36
3	102.51	102.79
4	98.24	97.56
5	100.14	100.54
6	104.14	104.08
7	114.85	115.07
8	128.13	128.52
9	123.39	123.12
10	132.76	132.97
11	127.50	126.91
12	134.77	135.46
13	140.88	140.54
14	140.89	140.71
15	146.09	146.44
16	142.38	142.72
17	148.85	148.97
18	157.93	158.30
19	146.44	146.46
20	143.32	143.70
21	143.90	143.43
22	148.31	148.17
23	150.14	149.78
24	159.33	158.88

Mes	y	f(x)
25	164.65	165.02
26	174.63	174.98
27	181.87	182.11
28	189.78	190.36
29	199.36	199.04
30	209.82	209.57
31	199.27	198.41
32	195.76	195.71
33	198.72	198.95
34	190.30	190.59
35	188.57	188.76
36	182.74	182.64
37	186.25	185.93
38	183.92	183.44
39	188.29	187.63
40	195.77	196.17
41	201.67	202.04
42	197.54	197.25
43	185.08	185.02
44	169.95	169.43
45	177.91	178.09
46	185.48	185.22
47	186.91	186.00
48	183.14	183.84
49	180.57	180.79

Tabla 5.21: Análisis de la función obtenida en el ajuste del índice de precios general de mercancías

Materia prima	Categoría	Importancia	Correlación
Aceite de colza	Aceites y harinas	84.29%	0.87
Cerdo	Carnes	3.19%	0.73
Café robusta	Bebidas	2.73%	0.75
Pulpa de celulosa	Materias primas agrícolas	1.03%	0.67

El experimento se repitió en 30 ocasiones con un error cuadrático medio de 0.4 y un coeficiente de determinación de 0.9999.

5.12. Comparación de resultados

En esta sección vamos a mostrar los ajustes de curvas que se pueden generar con un método alternativo al nuestro como lo es MATLAB. Este software utiliza herramientas deterministas para resolver ajustes a diferencia de la heurística que proponemos en esta tesis.

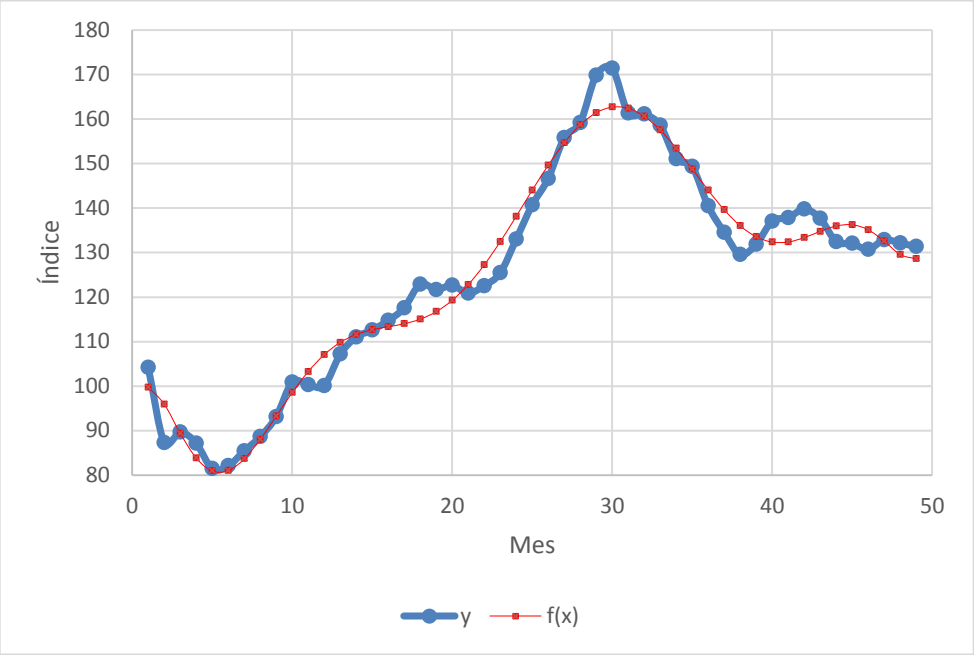

Figura 5.10: Ajuste del índice de precios de materias primas agrícolas generado por MATLAB

Todos los ajustes mostrados en esta sección fueron realizados con la herramienta *Curve Fitting* de MATLAB y usaremos como modelo un polinomio de grado nueve, el de más alto grado que permite el software.

Se muestra en la Figura 5.10 el ajuste del índice de precios de materias primas agrícolas. El error cuadrático medio obtenido fue de 4.17.

El ajuste del índice de precios de alimentos se muestra en la Figura 5.11. El error cuadrático medio obtenido fue de 4.02.

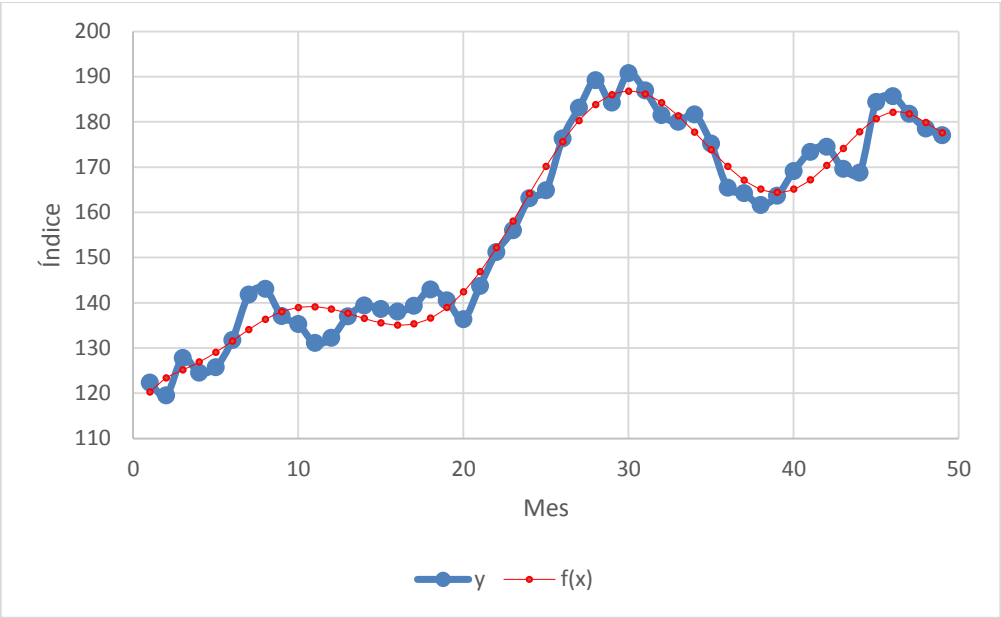

Figura 5.11: Ajuste del índice de precios de alimentos generado por MATLAB

El ajuste del índice de precios de bebidas se muestra en la Figura 5.12. El error cuadrático medio obtenido en éste ajuste fue de 4.09.

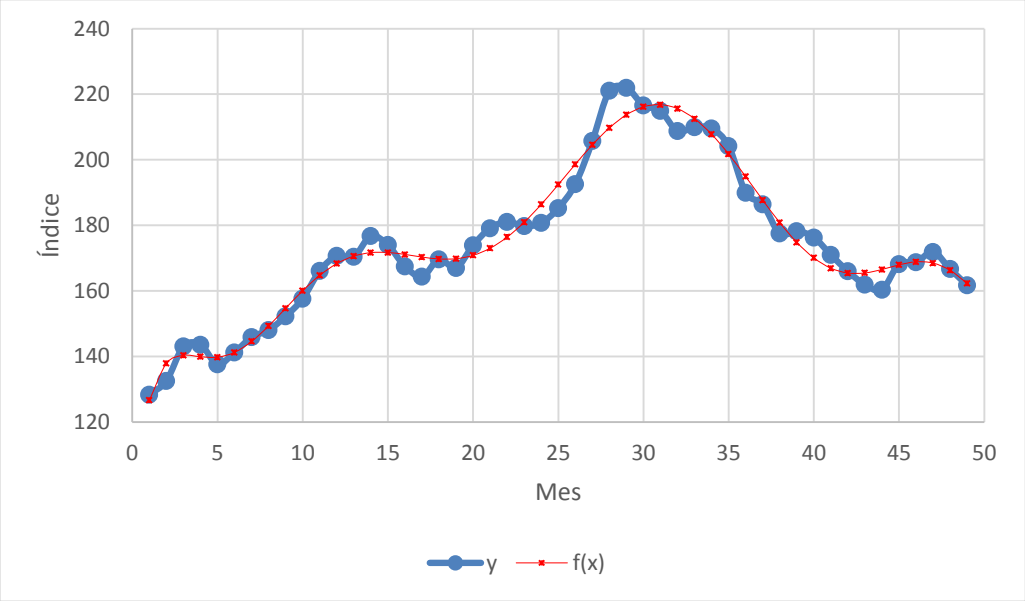

Figura 5.12: Ajuste del índice de precios de bebidas generado por MATLAB

El ajuste del índice de precios de combustibles se muestra en la Figura 5.13. El error cuadrático medio obtenido fue de 8.45.

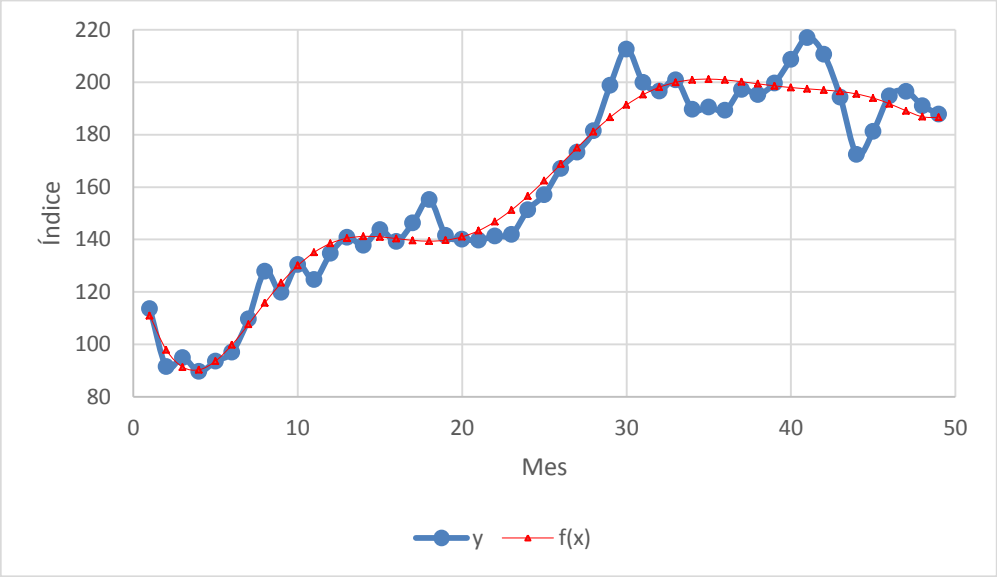

Figura 5.13: Ajuste del índice de precios de combustibles generado por MATLAB

El ajuste del índice de precios de insumos industriales se muestra en la Figura 5.14. El error cuadrático medio obtenido fue de 6.69.

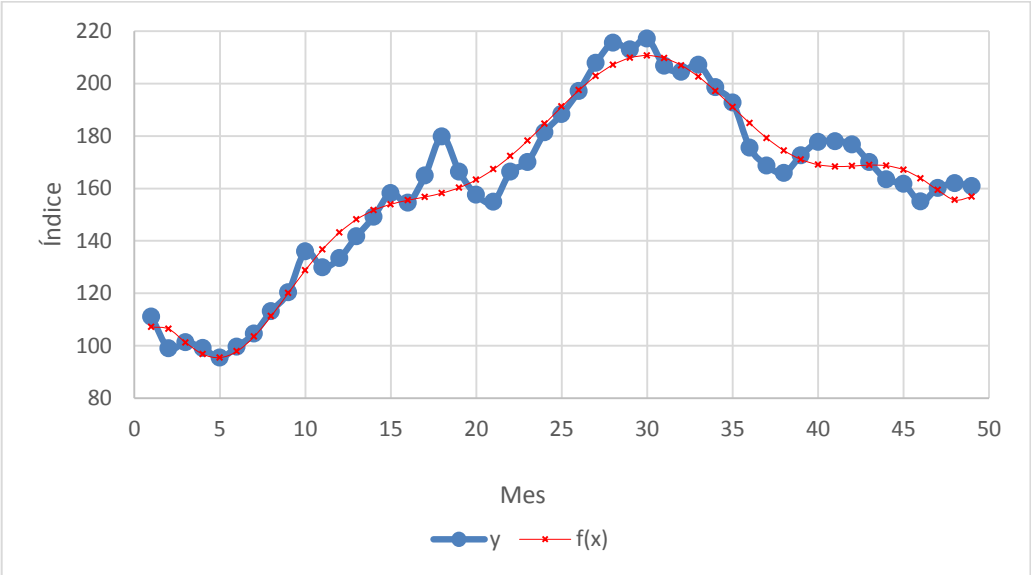

Figura 5.14: Ajuste del índice de precios de insumos industriales generado por MATLAB

El ajuste del índice de precios de alimentos y bebidas se muestra en la Figura 5.15. El error cuadrático medio fue de 3.77.

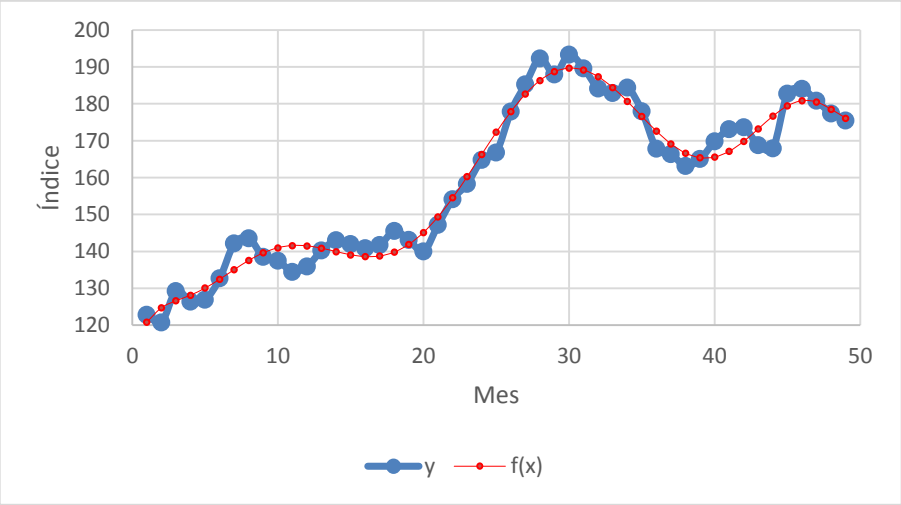

Figura 5.15: Ajuste del índice de precios de alimentos y bebidas generado por MATLAB

El ajuste del índice de precios de petróleo crudo se muestra en la Figura 5.16. El error cuadrático medio obtenido fue de 9.69.

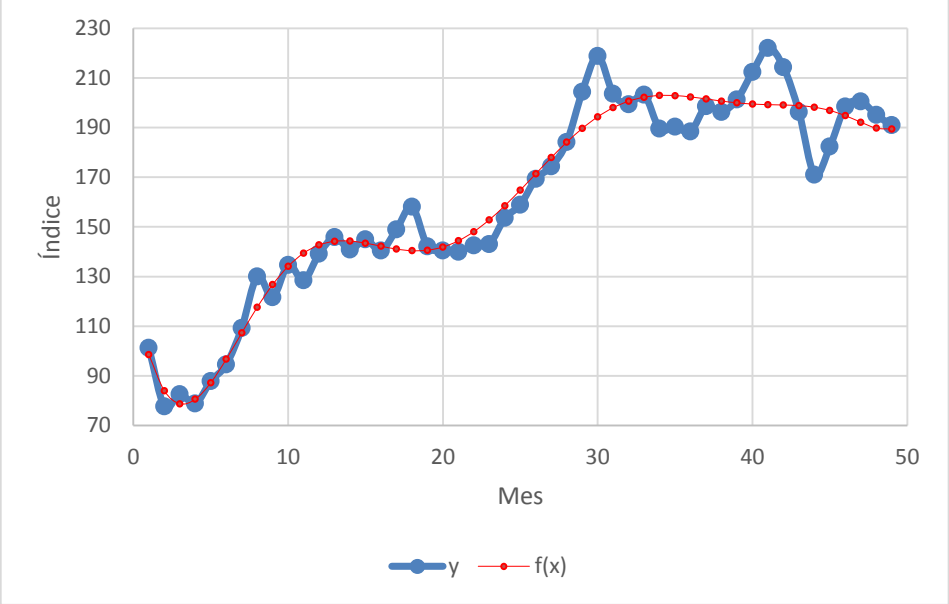

Figura 5.16: Ajuste del índice de precios de petróleo crudo generado por MATLAB

El ajuste del índice de precios de no combustibles se muestra en la Figura 5.17. El error cuadrático medio obtenido fue de 4.69.

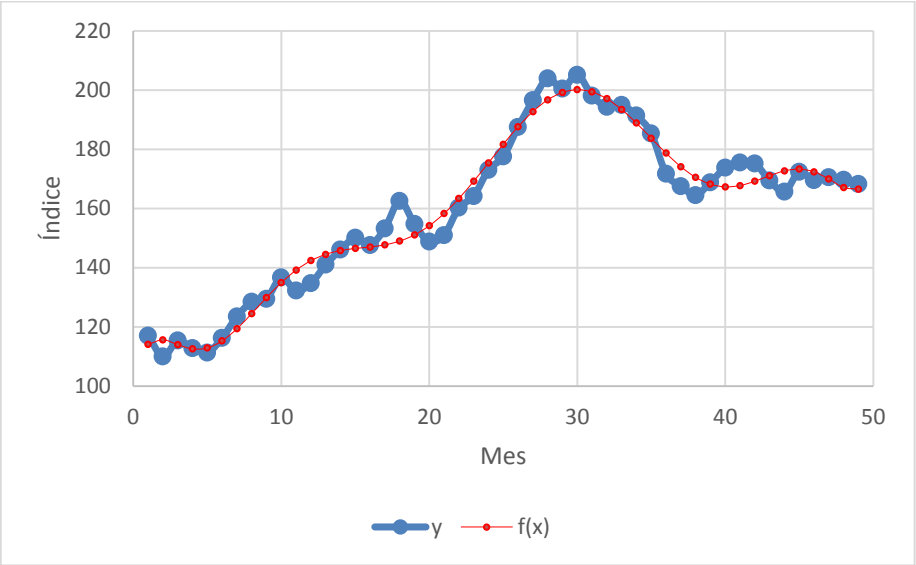

Figura 5.17: Ajuste del índice de precios de no combustibles generado por MATLAB

El ajuste del índice de precios general de mercancías se muestra en la Figura 5.18. El error cuadrático medio obtenido fue de 6.64.

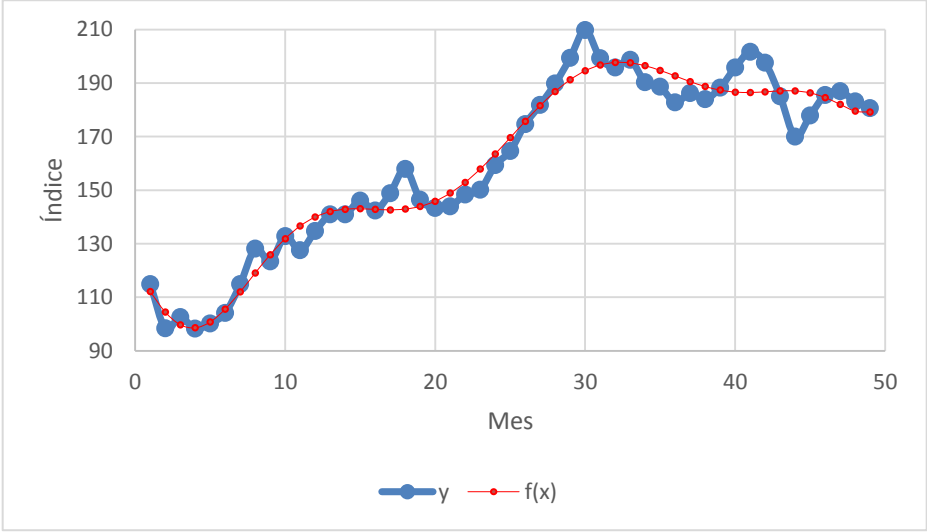

Figura 5.18: Ajuste del índice de precios general de mercancías generado por MATLAB

En la Tabla 5.22 se comparan los resultados obtenidos por MATLAB y los resultados obtenidos por el sistema de programación genética GP Curve Fitting desarrollado en esta investigación (Capítulo IV).

Tabla 5.22: Comparación de resultados entre MATLAB y GP Curve Fitting. El error cuadrático medio de GP Curve Fitting es obtenido del promedio de la ejecución de 30 experimentos

Ajuste	Error cuadrático medio MATLAB	Error cuadrático medio GP Curve Fitting
Índice de precios de materias primas agrícolas	4.17	0.3
Índice de precios de alimentos	4.02	0.4
Índice de precios de bebidas	4.09	0.7
Índice de precios de combustibles	8.45	0.8
Índice de precios de insumos industriales	6.69	1.3
Índice de precios de alimentos y bebidas	3.77	0.4
Índice de precios de petróleo crudo	9.69	0.4
Índice de precios de no combustibles	4.69	0.4
Índice de precios general de mercancías	6.64	0.4

CAPÍTULO VI

CONCLUSIONES Y TRABAJO FUTURO

Contenido

Este capítulo contiene una explicación de los aspectos de la investigación que resultaron correctos o incorrectos de acuerdo a los objetivos propuestos. También algunas ideas para poder extender el proyecto a trabajos futuros.

6.1 Conclusiones

Modelar la información siempre ha sido una forma de ayudar al humano a comprender un fenómeno que es demasiado complejo.

Durante el desarrollo de esta investigación hemos generado varios modelos económicos, como el ajuste de varias curvas de datos que son índices de algún insumo. Por ejemplo, el índice de precios de combustibles, el índice de precios de materias primas agrícola, etc. Estos modelos económicos son creados con un sistema de programación genética (PG) que permite incorporar a las funciones múltiples variables independientes. Dichas variables independientes representan datos que tienen relación con los puntos que se desean ajustar. Por ejemplo, el precio de la gasolina, precio del cuero, precio del petróleo.

Los ajustes de curvas resultaron razonablemente buenos considerando el enorme espacio de búsqueda del problema. Encontramos que para calcular un indicador económico no existe una única función o relación entre variables independientes que lo generen. Existen muchos caminos para llegar al resultado. El descubrir que existen 2 o más formas de relacionar variables independientes para producir una función matemática que describa una serie de puntos dados, es una parte importante e imprevista de la investigación. No obstante, debilita una parte que deseábamos incorporar al trabajo. Esta es, el análisis de las funciones matemáticas. Debido a que no existe una única relación entre las variables para crear un indicador económico en particular y tomando en cuenta que en un sistema de programación genética no tenemos control total sobre las soluciones, éstas se crean de manera estocástica. Entonces, el análisis de las propiedades de la función será sólo para esa función en específico. Si se decide ejecutar el programa para generar una nueva solución es muy probable que el resultado del análisis cambie abruptamente. Por esa razón no podemos asegurar qué variables serán importantes para ajustar los datos de un indicador económico de manera general.

No obstante, un modelo económico no tiene una única utilidad; puede ser usado para realizar predicciones sobre el comportamiento de los hechos y determinar los efectos o tomar decisiones sobre los mismos.

El tener datos relacionados a la curva que se va a ajustar (series de datos que cambian con respecto al tiempo en que se ejecutan) ha demostrado ayudar en la creación de un mejor ajuste. Esto es, si lo que estamos buscando es una función que simule el comportamiento de un fenómeno económico general, no es aventurado pensar que existen otros fenómenos económicos particulares que pueden ser relacionados entre ellos y con ciertas constantes, de tal forma que se genere un comportamiento similar al del fenómeno general.

En resumen, creamos un sistema de programación genética en LISP que resuelve ajustes de curva con múltiples variables y que es una mejora a los sistemas de ajuste de una sola variable, con el cual analizamos un conjunto de datos de tipo económico (ver capítulo 5). Con ello, hemos cumplido los objetivos generales y específicos que se plantearon al inicio de la presente investigación.

Llegamos a varias conclusiones importantes entre ellas; dejamos de asumir que la correlación tenga una relación directa con la relevancia de cada variable independiente; también hemos visto casos donde las variables que se consideran relevantes para la función matemática creada, no pertenecen a la categoría o categorías que abarca el indicador económico. Esto es debido a que la PG puede generar, en algunas ocasiones, soluciones hasta el momento desconocidas

6.2 Sobre usos del sistema de programación genética "GP Curve Fitting"

Como parte del proceso de delimitación de la investigación, centramos nuestros esfuerzos en generar ajuste de curvas sobre datos de tipo económico. Sin embargo, el sistema GP Curve Fitting es capaz de recibir datos de cualquier índole. Por ejemplo, hablando en términos de biología, podemos ajustar una curva que represente el crecimiento de un árbol en diez años (reportando el crecimiento cada mes). Entonces las variables independientes que ayuden a la creación podrían ser las propiedades del suelo, por ejemplo, textura, estructura, permeabilidad, porosidad, drenaje, profundidad efectiva, consistencia, etc., que también deberán ser muestreadas por mes.

Por otra parte, si el ajuste no requiere que se usen variables independientes, es decir, que se requiera un ajuste sencillo que use sólo constantes aleatorias para generar cierta función, es posible no introducir variables independientes.

6.3 Sobre el trabajo futuro

Existen algunas ideas que se proponen para investigaciones futuras. Una propuesta es que el sistema de programación genética cuente con un proceso interno que evalúe por intervalos los datos que se van a ajustar; de esta forma se crearían varias funciones para el ajuste de una sola curva. Es una forma de divide y vencerás, en donde funciones más pequeñas en ciertos intervalos de tiempo pueden generar un ajuste más complejo.

Se propone experimentar con predicción de datos; hacer pruebas de este tipo podría mostrar si nuestro sistema predice mejor los datos que los algoritmos actuales.

Referencias

Ahn, C. W. y Ramakrishna, R. S., 2002. *A genetic algorithm for shortest path routing problem and the sizing of populations*. IEEE Transactions on Evolutionary Computation, 6(6), páginas 566–579.

Ahn, C. W., 2006. *Advances in Evolutionary Algorithms, Theory, Design and Practice*. Nueva York. Springer, página 175.

Ahn, C. W., Ramakrishna, R. S., Kang, C. G., y Choi, I. C., 2001. *Shortest path routing algorithm using hopfield neural network*. Electronics Letters, 37(19), páginas 1176–1178.

Angeline, P. J., 1998. *A historical perspective on the evolution of executable structures.* Páginas:179–195.

Ashlock, D., 2005. *Evolutionary Computation for Modeling and Optimization.* Canada, Springer.

Back, T. y Schwefel, H., 1993. *An overview of evolutionary algorithms for parameter optimization*. Evolutionary Computation, 1, 1-23.

Banzhaf, W., Nordin, P., Keller, R. E., y Francone, F. D., 1998. *Genetic Programming: An Introduction.* Morgan Kaufmann, San Francisco.

Bosman, P. A. N., 2003. *Design and application of iterated density-estimation evolutionary algorithms*. Tesis doctoral, Universidad de Utrecht, TB Utrecht, Holanda.

Coello, C. A., B. Lamont, G. y Van Veldhuizen, D. A., 2007. *Evolutionary Algorithms for Solving Multi-Objetive Problems.* 2da edición. Springer.

Cramer, N. L., 1985. *A representation for the adaptive generation of simple sequential programs.* Primera conferencia internacional de algoritmos genéticos y sus aplicaciones, páginas 183–187.

De Jong A.K., 1975, *An analysis of the behavior of a class of genetic adaptive systems*. Tesis doctoral. Universidad de Michigan.

Fogel, L. J., Owens, A. J., y Walsh, M. J., 1966. *Artificial Intelligence through Simulated Evolution*. John Wiley & Sons, Nueva York, Estados Unidos de Norte América.

Forsyth, R. 1989. *The evolution of intelligence. Machine Learning, Principles and Techniques*., capítulo 4, páginas 65–82. Chapman and Hall.

Forsyth, R. y Rada, R. 1986. *Machine Learning applications in Expert Systems and Information Retrieval*. Ellis Horwood series in Artificial Intelligence. Ellis Horwood, Chichester, Reino Unido.

Forsyth, R., 1981. *BEAGLE a Darwinian approach to pattern recognition*. Kybernetes, 10:159–166.

Friedberg, R. M., 1958. *A learning machine: Part I*. IBM Journal of Research and Development, 2:2–13.

Friedberg, R. M., Dunham, B., y North J. H., Marzo 1959. *A learning machine: Part II*. IBM Journal of Research and Development, 3(3):282–287.

Friedman, J., 1988, *Multivariate adaptive regression splines*. Departamento de Reportes Técnicos Estadísticos, Universidad de Stanford.

Fujuki, C., 1986. *An evaluation of Holland's genetic algorithm applied to a program generator*. Tesis de maestría, Departamento de ciencias de la computación, Universidad de Idaho, Moscow, Estados Unidos de Norte América.

Goldberg, D. E., 1989. *Genetic algorithms in search, optimization, and machine learning*. MA: Addison-Wesley.

Gulsen, M., Smith, A. E., y Tate, D. M., 1995. *A genetic algorithm approach to curve fitting*. Int. J. Prod. Res., Departamento de ingeniería industrial. Universidad de Pittsburgh, Estados Unidos de Norte América.

Heitkotter, J. y Beasley, D., 1998. *Guide to Evolutionary Computation: A List of Frequently Asked Questions (FAQ)*. ENCORE (The EvolutioNary Computation REpository Network).

Hicklin, J. F., 1986. *Application of the genetic algorithm to automatic program generation.* Tesis de maestría, Departamento de ciencias de la computación, Universidad de Idaho, Estados Unidos de Norte América.

Kamal, H. A. y Eassa, M. E., 2002. *Solving curve fitting problems using genetic programming*. IEEE MELECON. El Cairo, Egipto.

Karp, R. M., 1972, *Reducibility among combinatorial problems*. En R. E. Miller J. W. Thatcher. Complexity of computer computations. Nueva York: Asamblea. pp.85-103.

Karr, C. L., Stanley, D. A., y Scheiner, B. J., 1991, *Genetic algorithm applied to least squares curve fitting.* U.S. Bureau of Mines. Reporte de investigaciones 9339.

Kinnear, K. y Angeline, P., 1996. *Advances in Genetic Programming, Volumen 2*. The MIT Press, Cambridge, MA.

Kinnear, K., 1994. *Advances in Genetic Programming*. The MIT Press, Cambridge, MA.

Koza, J. R., 1988. *Non-Linear Genetic Algorithms for Solving Problems*. United States Patent and Trademark Office, patente número 4,935,877.

Koza, J. R., 1989. *Hierarchical genetic algorithms operating on populations of computer programs*. En actas de la onceava conferencia internacional de Inteligencia Artificial IJCAI-89, páginas 768–774.

Koza, J. R., 1990a. *A hierarchical approach to learning the boolean multiplexer function.* En actas del primer taller sobre fundamentos de algoritmos genéticos, páginas 171–191.

Koza, J. R., 1990b. *Evolution and co-evolution of computer programs to control independent-acting agents*. En actas de la primera conferencia internacional sobre simulación de comportamiento adaptivo, páginas 366–375.

Koza, J. R., 1990c. *Genetic evolution and co-evolution of computer programs*. En Artificial Life II, páginas 603–629.

Koza, J. R., 1990d. *The genetic programming paradigm: Genetically breeding populations of computer programs to solve problems.* Reporte técnico STAN-CS-90-1314, Departamento de ciencias de la computación, Universidad de Stanford, Margaret Jacks Hall, Estanford, CA 94305.

Koza, J. R., 1992a. *Genetic Programming*. The MIT Press, Cambridge, MA.

Koza, J. R., 1992b. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. A Bradford Book. The MIT Press, Cambridge, Massachusetts, Estados Unidos de Norte América.

Koza, J. R., 1992c. *Non-Linear Genetic Algorithms for Solving Problems by Finding a Fit Composition of Functions*. United States Patent and Trademark Office, patente número 4,935,877.

Koza, J. R., 1994. *Genetic Programming II*. The MIT Press, Cambridge, MA.

Koza, J. R., 1999. *Genetic Programming III*. Morgan Kaufmann, San Francisco.

Messa, K. y Lybanon, M. 1991. *Curve Fitting Using Genetic Algorithms*. Universidad Loyola, departamento de ciencias matemáticas, Nueva Orleáns.

Muhlenbein, H., y Schlierkamp-Voosen, D., 1993. *Predictive models for the breeder genetic algoritm: Continuous parameter optimization*. Evolutionary Computation, 1, 25-49.

Pelikan, M., 2002. *Bayesian optimization algorithm: From single level to hierarchy*. Tesis doctoral, Universidad de Illinois en Urbana-Champaign, Urbana, Illinois.

Rogers, D., 1991. *A hybrid of Friedman's multivariate adaptive regression splines (MARS) algorithms with Holland's genetic algorithm.* En actas de la cuarta conferencia internacional de algoritmos genéticos, páginas 384-391.

Samuel, A. L., 1959. *Some studies in machine learning using the game of checkers*. IBM Journal of Research and Development, 3(3):210–229.

Senturk, S., 2009. *Applied genetic algorithms approach to curve fitting problems*. Tesis de maestría. Universidad de Bahcesehir. Estanbul, Turquía.

Smith, S. F., 1980. *A Learning System based on Genetic Adaptive Algorithms*. Tesis doctoral, Universidad de Pittsburgh, Pittsburgh, Estados Unidos de Norte América.

Spears, W., 1998. *Evolutionary Algorithms, The Role of Mutation and Recombination.* Universidad de George Mason, Virginia. Springer.

Sumathi, S., Hamsapriya, T. y Surekha, P., 2008 *Evolutionary Intelligence.* Coimbatore India.

Wetzel, A., 1983. *Evaluation of the effectiveness of genetic algorithms in combinatorial optimization.* Tesis doctoral, Universidad de Pittsburgh, Pittsburgh, Filadelfia, Estados Unidos de Norte América.