UNIVERSIDAD NACIONAL DE COLOMBIA

Tesis de Doctorado

Generación Automática de Modelos de Pronóstico usando Bloques Funcionales y Programación Genética

Carlos Alberto Martínez, MSc

Director:
Prof. Juan David Velásquez Henao, MSc, PhD

Universidad Nacional de Colombia
Facultad de Minas, Doctorado en Ingeniería–Ingeniería de Sistemas
Medellín, Colombia
2016

Resumen

En el marco de la predicción de series temporales, la Programación Genética ha tomado gran fuerza en los últimos años debido a su capacidad de deducir la ecuación y aquellos parámetros que mejor aproximan la relación entre la variable de salida y el conjunto de variables de entrada; sin embargo, al ser aplicada en la predicción de series de tiempo, aún presenta limitaciones en la incorporación de las componentes de ciclo, tendencia, estacionalidad y error; en el uso de aquellos rezagos de interés en todos los individuos durante el proceso de búsqueda; en la inclusión de los modelos bechmark de predicción de series de tiempo presentes en la literatura; y la redundancia de nodos (terminales y operadores) que no aportan a la aptitud del modelo. Para abordarlos, en este trabajo se modificaron: la estructura del algoritmo de programación genética original, la función de aptitud, los operadores de selección, intensificación, reproducción, mutación y cruce; además, fueron incorporadas las componentes de ciclo, tendencia, estacionalidad y error, a los bloques funcionales. Lo anterior permite la inclusión de las componentes de los modelos actuales de predicción de series de tiempo, la focalización de los individuos en regiones de interés durante el proceso de exploración, y la incorporación de conocimiento experto en la generación de la población inicial del algoritmo. Las modificaciones propuestas fueron implementadas en un prototipo en el lenguaje R, y validadas contra series de tiempo con ecuación de generación conocida (para verificar la capacidad de deducción de la ecuación a partir de los datos) y series benchmark de la literatura de predicción de series de tiempo, como son las series: AIRLINE, SUNSPOT, LYNX, INTERNET y POLLUTION. Los resultados obtenidos en términos de medidas de error comparados contra modelos ARIMA, SVM (Maquinas de vectores de soporte), MLP (perceptrones multicapa), NN (redes neuronales artificiales), DAN (redes neuronales de arquitectura dinamica) y el algoritmo original de programación genética, fueron mejores tanto en el entrenamiento como la predicción.

Palabras clave: Pronóstico; Series de tiempo; Regresión simbólica; Programación genética; Modelos matemáticos.

Abstract

In the framework of time-series forecasting, Genetic Programming has taken great strength in recent years due to their ability to derive the equation and the parameters that best approximate the relationship between the output variable and the set of input variables; but when applied to the prediction time series, is still limited in the incorporation of cycle, trend, seasonality and error components; in the use of lags of interest in all individuals during the search process; in the inclusion of bechmark models of literature of time series forcasting, and the redundancy of nodes (terminals and operators) that do not contribute to the fitness of the model. To address them, in this work were modified the structure of the original genetic programming algorithm, the fitness function, selection operators, intensification, reproduction, mutation and crossover, in addition, it included cycle components, trend, seasonality and error, to the functional blocks. This allows the inclusion of components of current models of time series forecasting, the targeting of individuals in regions of interest during the exploration process, and the incorporation of expert knowledge in the generation of the initial population of the algorithm. The proposed changes were implemented in a prototype in the R language, and validated against time series generation equation with known (to verify deductibility of the equation from the data) and bechmarks of series of time series forecasting, such as the series: AIRLINE, SUNSPOT, LYNX, INTERNET and POLLUTION. The results in terms of error measures compared with ARIMA models, SVM (support vector machines), MLP (multilayer perceptron), NN (artificial neural network), DAN (Dynamic Architecture for Artificial Neural Networks) and original genetic programming algorithm, were both better training and prediction.

Keywords: Forecasting; Forecast; Time series; Symbolic regression; Genetic programming; Mathematical models**.**

Contenido

Pág.

Lista de Figuras

Pág.

Lista de Tablas

Pág.

Lista de símbolos y abreviaturas

Símbolo	Término
$f(.)$	Función generadora de la serie de tiempo.
$\mathbf{x}$	Vector de rezagos considerados.
$\boldsymbol{\theta}$	Vector de parámetros/coeficientes del individuo.
$f^*(.)$	Aproximación de $f(.)$.
δ	Error de aproximación entre $f(.)$ y $f^*(.)$.
$\boldsymbol{\lambda}$	Vector de coeficientes de las componentes de la serie de tiempo.
$S(.)$	Componente de estacionalidad.
$C(.)$	Componente de ciclo.
$\Gamma(.)$	Componente de tendencia.
$\Upsilon(.)$	Componente de error.
$B_i(.)$	Bloque funcional i.
$g_i(\cdot)$	Funciones generadoras del bloque funcional.
α_i	Coeficientes de $g_i(\cdot)$ tal que $B(.) = \alpha_0 + \sum_{i=1} \alpha_i G_i(.)$.
n	Tamaño de la población de individuos P_g.
g	Generación actual
m	Tamaño del vector de parámetros $\boldsymbol{\theta}$
p	Número de rezagos a ser considerados
q	Número de diferencias a ser consideradas
d	Nivel de diferenciación a ser considerado
I	Individuo de programación genética
P_g	Población individuos $I_i$de la generación g, con $1 \leq i \leq n$
T	Conjunto de terminales
l	Tamaño del conjunto de terminales
Z	Conjunto de operadores
k	Tamaño del conjunto de operadores
ζ_i	Operador i de Z tal que $\forall_i\ \zeta_i \in \mathrm{Z}$
y_t	Salida de la función $F(.)$ para el tiempo t
t	Tiempo actual
$\varkappa_{max}$	Profundidad máxima permitida en un individuo
$\varkappa$	Profundidad de un individuo, equivalente al número de niveles en su representación de árbol sintáctico
h	Número de nodos de un individuo
h_{max}	Número máximo de nodos permitidos en un individuo
β	Vector de probabilidades de selección de una componente con $\beta = \{\beta_1, \dots, \beta_4\}$
β_i	Componente i del vector β. $\beta_i \in \mathbb{R} \mid \forall_{1 \leq i \leq 4}\ 0 \leq \beta_i \leq 1$
1STP	Probabilidad de transición de un paso
AA	Método aritmético afín
AA+	Hibridación entre el método aritmético afin y el algoritmo de programación genética
ABC	Algoritmo de colonia de abejas

Símbolo	Término
ANN	Red neuronal artificial
ACO	Algoritmo de colonia de hormigas
BSTGP	Algoritmo de poda de individuos
CGP	Programación genética cartesiana
DDL	Algoritmo de limitación dinámica de profundidad
DE	Evolución diferencial
DGP	Algoritmo de programación genética distribuido
dMSXF	Método determinista de cruce de fusión multipaso
DPV	Algoritmo de variación dinámica de la población
EDS	Método de simplificación de decisión equivalente
EE	Estrategias Evolutivas
EM	Algoritmo de maximización de la expectativa
EPR	Regresión polinómica evolutiva
GA	Algoritmos genéticos
GA-P	Hibridación del algoritmo de programación genética y de algoritmos genéticos
GEP	Programación de expresiones genéticas
GGPA	Hibridación de los algoritmos de programación genética y algoritmos genéticos
GP	Programación genética
GPEP	Programación genética de genes
HDN	Nodos de alta densidad
HEA	Índice Hoeffding basado en algoritmos evolutivos
HVES	Algoritmo de separación de hiper-volúmenes
IC	Inteligencia computacional
IPT	Poda iterativa por torneo
KPCA	Kernel de análisis de componentes principales
KLDA	Kernel de análisis del discriminante lineal
MLP	Perceptrón multicapa
MSTP	Probabilidad de transición multipaso
MSXF	Método de cruce de fusión multipaso
MwM-GP	Algoritmo de programación genética de memoria con memoria
NN	Red neuronal artificial
NN-GP	Hibridación del algoritmo de redes neuronales y el de programación genética
OP	Optimización ordinal
OPTIM	Algoritmo de optimización basado en gradiente presente en el lenguaje R
OSL	Minimos cuadrados ortogonales
PTGP	Algoritmo de programación genética con estructura de punto de árbol
RGNOUD	Implementación de los algoritmos genéticos en el lenguaje R
SA	Algoritmo de recocido simulado
SAC	Método de cruce semántico consciente
SETAR	Modelo autorregresivo con umbral auto-excitable
SLP	Algoritmo de programa de linea recta
SR	Regresión simbólica
SRM	Minimización del riesgo estructural
SSC	Método de cruce basado en la similaridad semantica
SVM	Maquina de vector de soporte
S-Expression	Expresiones simbólicas
TAG	Árboles sintácticos contiguos

Símbolo	Término
TGP	Programación genética ruteada
TSPA	Algoritmo de perturbación de series de tiempo

1. Pronóstico de Series de Tiempo con Programación Genética: Estado Actual y Retos Futuros

1.1 Introducción

La necesidad de contar con herramientas orientadas a la toma adecuada de decisiones, ha conducido en las últimas décadas a un creciente interés por el uso de técnicas de inteligencia computacional en el desarrollo de modelos de pronóstico de series de tiempo [1], debido principalmente a su flexibilidad para aprender relaciones no lineales desconocidas; estas técnicas incluyen las redes neuronales artificiales [2], los sistemas de inferencia borrosa [2], los sistemas neurodifusos y las técnicas evolutivas como la programación genética [3]. La programación genética (GP, por su sigla en inglés) puede entenderse como la aplicación de los algoritmos genéticos a individuos que representan expresiones matemáticas [3]. Para ello, los individuos son representados como árboles sintácticos donde las funciones y operadores matemáticos (suma, resta, multiplicación, etc.) van en los nodos interiores (o nodos funcionales), y las variables y constantes numéricas van en los nodos terminales. De esta forma, al aplicar los operadores genéticos de clonación y cruce sobre una población de individuos, se realiza la búsqueda —en un espacio de expresiones matemáticas— que tiene como fin encontrar una ecuación empírica que permita representar la relación existente entre un conjunto de variables de entrada y una variable de salida; en este caso, la GP recibe el nombre de regresión simbólica (SR, por su sigla en inglés) [3].

La GP es una metodología robusta en el manejo de información faltante, ruido y determinación de las variables relevantes para la modelación de la relación entre los datos; converge a una función dentro del espacio de búsqueda, que representa, con alguna precisión, la relación entre las variables de entrada y las variables de salida, que de acuerdo con la literatura es mejor en términos de menor medida de error de predicción que las técnicas tradicionales; en teoría, el algoritmo genético converge a la mejor función obtenible que aproxima la relación entre los datos de una manera estructurada, siendo capaz de descartar variables irrelevantes o de poca importancia, tal que, selecciona aquellas con mayor poder explicativo del comportamiento de la variable dependiente.

Realizando una búsqueda en SCOPÚS con la cadena *TITLE-ABS-KEY (genetic programming OR symbolic regression) AND PUBYEAR > 1991 AND PUBYEAR < 2017* se reportan más de 25.431 artículos, muchos de los cuales corresponden a aplicaciones de la GP a la predicción de series de tiempo; por ejemplo, Kaboudan [4] usa la GP para la predicción de los precios diarios de acciones; Wang et al. [5] comparan la GP con varias técnicas de inteligencia artificial en el pronóstico de series de tiempo hidrológicas y concluyen que la GP podría superar a dichas técnicas alternativas en términos de su precisión; Abdelmalek et al. [6] utilizan la GP para obtener modelos de pronóstico de la volatilidad del índice S&P500. Sin embargo, y a pesar del gran volumen de literatura existente, los desarrollos en los siguientes aspectos son apenas incipientes:

- Cambios de estructura y componentes del algoritmo de GP para la mejora de los problemas de convergencia: garantizar que el algoritmo genético encuentre funciones con un desempeño mejor que los modelos comúnmente usados en series de tiempo.
- Factibilidad: que la solución encontrada posea consistencia matemática.

- Parsimonia: la solución encontrada no posea una excesiva cantidad de nodos que no aportan información al modelo.
- Inclusión de conocimiento experto: obligar al algoritmo genético para que considere explícitamente dentro de los individuos de la población, modelos ampliamente conocidos y de uso difundido en la literatura relacionada con la predicción de series de tiempo, con su respectiva representación como árboles sintácticos.
- Focalización del espacio de búsqueda: obligar al algoritmo genético a que focalice el proceso de búsqueda sobre regiones más prometedoras que podrían contener modelos empíricos con un desempeño superior a los modelos tradicionales de series de tiempo.
- Inclusión de todas aquellas variables de relevancia: incluir en cada uno de los individuos generados aleatoriamente durante el proceso de búsqueda, aquellos rezagos realmente relevantes en la explicación de la relación entre los datos de entrada y la salida del modelo resultante.

Es de anotar que se han realizado esfuerzos en el manejo de las respectivas falencias, centrándose principalmente en la hibridación de GP con otras técnicas como las redes neuronales polinomiales [7], la generación jerarquica de los individuos [8], el cambio en las medidas de error [9] y la poda de subárboles sintácticos del individuo [10, 11]; sin embargo, dichas falencias persisten y generan una clara disminución en la calidad de la solución encontrada por el algoritmo de GP (error de predicción y tamaño del individuo).

El resto de este capítulo está organizado como sigue. En la Sección 1.2 se analiza el trabajo previo realizado en los estudios de maestría. En la Sección 1.3 se realiza una reexaminación de la literatura actual y se analizan las falencias encontradas. En la Sección 1.4 se plantean las preguntas emergentes a partir de la Sección 1.3. En la Sección 1.5 se plantea la hipótesis del trabajo de doctorado. En la Sección 1.6 se proponen los objetivos de la tesis, y en la Sección 1.7 se muestra la organización general del documento.

1.2 Motivación de este trabajo

La idea original de este trabajo surge de la necesidad de contar con algoritmos eficientes que realmente describan el comportamiento de las series de tiempo, para lo cual se propone el uso del algoritmo de programación genética, debido principalmente a su capacidad de relacionar la estructura y parámetros entorno a una función matemática llamada individuo, que encierra el comportamiento de la serie de tiempo analizada. Sin embargo, es necesaria una serie de modificaciones orientadas a una búsqueda más eficiente del algoritmo de GP, debido a que éste se basa en la búsqueda aleatoria de individuos en un espacio funcional infinito, definido por todas las posibles combinaciones entre los terminales y los operadores seleccionados.

Por lo anterior, en la tesis de maestria titulada *Problemas abiertos en la aplicación de la Regresión Simbólica en el pronóstico de series de tiempo* [12], se introdujo el concepto de bloques funcionales (BF) en el conjunto de terminales del algoritmo tradicional de programación genética, el cual permitio reducir —en parte— el tamaño de los individuos, al representar subárboles sintácticos por medio de un único nodo. Por ejemplo: el modelo Auto Regresivo propuesto por Box & Jenkins $\left(AR(\mathbf{x}, p)\right)$ [13], correspondería a un único bloque funcional $AR(\mathbf{x}, p) = \sum_{i=1}^{p} w_i x_i = B(\mathbf{x}, \mathbf{w})$, con $\mathbf{w} = [w_1, \dots, w_p]$ y x_i equivalente al rezago en el tiempo $t - i$ de **x,** pasando de una representación tradicional de $4p + 1$ nodos ($p - 1$ nodos de suma, p nodos de multiplicación, p nodos de parámetros y p nodos de rezagos) a solo un nodo (equivalente al bloque funcional). Lo anterior fue validado en cinco series benchmark de la literatura en predicción de series de tiempo con resultados de error de predicción inferior a los estudios previos realizados por medio de modelos de redes neuronales, modelos ARIMA y maquinas de vectores de soporte (SVM, por su sigla en inglés).

En las siguientes subsecciones son ampliadas las modificaciones desarrolladas en la tesis de maestría [12], las cuales constituyen el punto inicial de análisis de los problemas y posibilidades de mejora planteados en este trabajo de doctorado.

1.2.1 Programación genética

La programación genética puede entenderse como la generalización de los algoritmos genéticos, en la cual los individuos representan expresiones matemáticas por medio de árboles sintácticos [3]. El objetivo de GP es encontrar aquel individuo que represente la ecuación empírica que permita representar la relación entre un conjunto de variables de entrada y una variable de salida.

Para lograr lo anterior, el algoritmo de GP utiliza los siguientes pasos:

1. Se genera una población inicial de $P_{g=0}$ de n individuos.
2. Para cada individuo de la población se calcula su valor de aptitud.
3. Se genera una nueva población P_g^* aplicando los operadores genéticos de cruce y mutación.
4. Se reemplaza la población actual por la población de hijos generada $P_g = P_g^*$.
5. Se evalúan los criterios de parada (usualmente el número máximo de generaciones), si no se cumplen se vuelve al paso 2. En caso contrario termina la ejecución del algoritmo.

Para la realización de los pasos del algoritmo de GP son necesarios las siguientes componentes:

- *Terminales:* Corresponde al vector de entradas de los individuos del algoritmo de GP, equivalente a los nodos hoja del árbol sintáctico, suele ser representado por T, y contiene las constantes y variables ($\mathbf{x}$) a ser empleadas para la generación de los individuos [3] (Figura 1-1).
- *Operadores o Funcionales:* Es el vector de operadores a ser utilizados en la generación de los individuos, corresponde a los nodos interiores del árbol sintáctico, y es representado por Z. Por ejemplo, el conjunto de operadores aritméticos básicos $Z = \{+, -, *, /\}$ [3] (Figura 1-1). Es de resaltar que cada operador debe estar en la capacidad de calcular cualquier tipo de entrada (inclusive erróneas), por lo que es necesaria su redefinición durante la implementación del algoritmo de GP para cumplir la propiedad de cierre [13].
- *Individuo:* Es la combinación de operadores y terminales organizados de manera jerárquica por medio de un árbol sintáctico (con un número de nodos H y una profundidad o número de niveles L), equivalente a una ecuación matemática [3, 14]. Se representa por I (Figura 1-1).
- *Población:* Corresponde al conjunto de n individuos en una generación g. Se representa por P_g [3].
- *Generación:* Corresponde a la iteracción especifica del algoritmo de GP. Se representa por g [3].
- *Función de aptitud:* Es la función que califica, por medio de un número real (valor de aptitud), que tan aceptable es el individuo; suele utilizarse medidas de error de predicción basados en distancia y variabilidad. Algunas propuestas realizadas por Koza [3] corresponden a: la función de aptitud cruda (suma de los errores por fuera del rango de aptitud permitido), la función de aptitud estandarizada (valor absoluto de la diferencia entre el valor esperado y valor de individuo), y la función de aptitud ajustada (función de aptitud estandarizada reescalada entre 0 y 1).
- *Operadores genéticos:* Son los responsables de diversificar la búsqueda. Existen dos operadores genéticos: el operador genético de clonación y el operador genético de cruce. El operador genético de clonación consiste en seleccionar un individuo y generar una copia idéntica de él (Figura 1-2); la selección del individuo se basa en el valor de aptitud de los individuos de la población por medio de alguna de las siguientes técnicas: proporcional a la aptitud, selección por rango, y selección por torneo [3]. El operador genético de cruce consiste en seleccionar dos individuos de la población (individuos padre), a los cuales se le selecciona un nodo de cruce

aleatorio desde el cual se intercambian los subárboles, generando dos individuos nuevos llamados hijos [3, 14] (Figura 1-3).

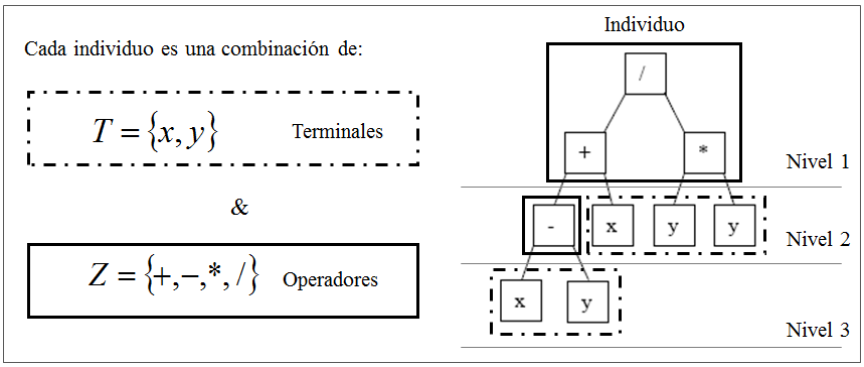


Figura 1-1: Diagrama en forma en árbol sintáctico de la ecuación: $((x - y) + x)/y^2$ con número de nodos $H = 9$ y profundidad $L = 3$.

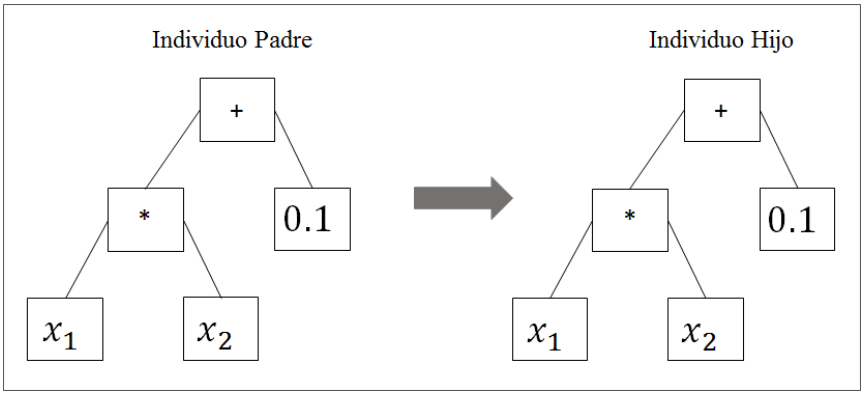


Figura 1-2: Operador genético de clonación.

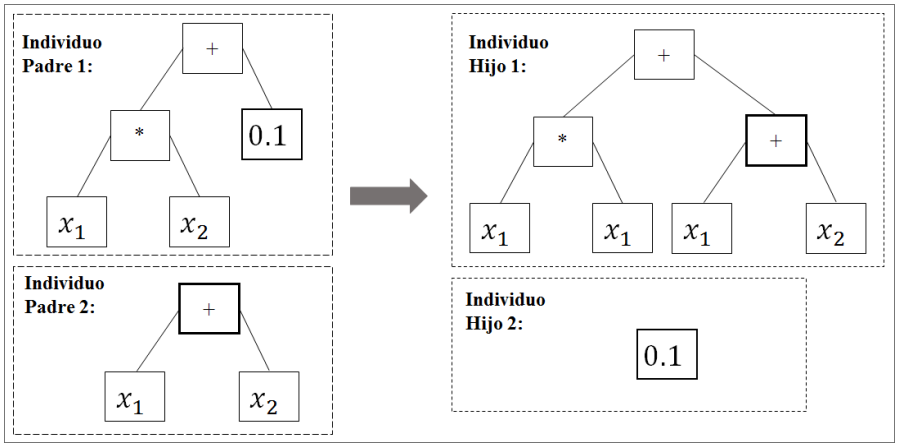


Figura 1-3: Operador genético de cruce.

1.2.2 Bloques funcionales

Los bloques funcionales corresponden a una función $B(.)$ la cual puede ser evaluada numéricamente sin depender de otras funciones externas. Un bloque funcional puede entenderse como una combinación lineal de las funciones $G_i(.)$, tal que: $B(\mathbf{x}, \mathbf{w}) = w_0 + \sum_i w_i G_i(\mathbf{x})$, donde **x** es el vector de entradas (variables/rezagos) del modelo, **w** corresponde al vector de parámetros y pesos de la combinación lineal de $G_i(.)$, y w_0 equivalente al intercepto. Lo anterior implica que es posible recombinar bloques funcionales igualando $G_i(.)$ a un bloque funcional $B(.)$, logrando así, expresar distintos tipos de modelos en función de un número finito de bloques funcionales.

Entre las principales características de los BF se encuentran:

- Los BF son funciones completas, es decir, a partir de un vector de entradas **x** y parámetros **w** pueden generar un resultado númerico sin depender de funciones externas.
- Al retornar un valor real, es posible generar combinaciones lineales y no lineales entre bloques funcionales y otros terminales por medio de de un conjunto de operadores (Z) definidos.
- Es posible anidar bloques funcionales igualando $G_i(.) = B(.)$, permitiendo la generación de modelos más complejos e hibridación de los mismos, a partir de un conjunto reducido de bloques funcionales.

Al introducirlos en el conjunto de terminales T del algoritmo de GP, se disminuye el tamaño de los individuos y podría representar modelos matemáticos —específicamente de predicción de series de tiempo— durante el proceso de búsqueda de GP de una manera más directa.

1.2.3 Aportes y contribuciones previas

En la aplicación de la GP a la predicción de series de tiempo se han propuesto distintas mejoras al algoritmo, entre las que se incluyen: la modificación en la estructura de los individuos del algoritmo de programación genética propuesta en la tesis de maestría [12], en la que se concluyó que la inclusión de bloques funcionales en el conjunto de terminales mejoraban la capacidad de predicción del modelo resultante, para lo cual fue implementado un paquete de clases S4 [15] en R llamado gpTool, que incluía la modificación propuesta y su posterior validación contra cinco series benchmark de la literatura de predicción de series de tiempo, para las que gpTool registró menor error de predicción; Hoang et al. [8] presentaron una hibridación del algoritmo DEVTAG (Tree Adjoining Grammar Guided) con el algoritmo original de programación genética, permitiendo la generación jerárquica de los individuos a partir de un conjunto de terminales T y un conjunto de símbolos no terminales V donde $\{ T \cap V = \emptyset \}$; y Nikolaev & Iba [7] presentaron la hibridación del algoritmo original de GP con las redes neuronales artificiales (ANN, por su sigla en inglés) por medio de la reformulación de las distintas neuronas de la capa oculta como la suma de bloques polinomiales.

1.2.4 Validación númerica de las contribuciones previas

Entre las distintas modificaciones propuestas al algoritmo de GP se resalta la propuesta realizada en la tesis de maestria de Martínez [12], la cual registró el menor error predicción en la predicción de las series de tiempo: AIRLINE, LYNX, POLLUTION, INTERNET y SUNSPOT, cuyos resultados se analizan a continuación:

- **Aproximación serie AIRLINE**
  La serie AIRLINE corresponde al número de pasajeros transportados mensualmente al exterior por una aerolínea entre Enero de 1949 (ENE1949) y Diciembre de 1960 (DIC1960); esta serie presenta una tendencia creciente y un patrón cíclico de periodicidad anual, los cuales exhiben un comportamiento no lineal. En [12] son presentados los resultados obtenidos al aplicar el

algoritmo gpTool, comparados con los resultados obtenidos en la literatura por medio de modelos ARIMA, MLP, SVM, ANN y el modelo original de GP.

Al realizar la comparación de la suma de errores cuadráticos (SSE, por su sigla en inglés, equivalente a sumatoria de las diferencias cuadráticas entre los resultados generados por el modelo propuesto y los valores reales registrados por la serie de tiempo), entre el modelo generado por medio del algoritmo gpTool y los demás modelos reportados en la literatura, se constata una mejora del 5% con respecto a los modelos SVM, del 97% con respecto a los modelos de ANN, del 97% con respecto a los modelos MPL, del 97% con respecto a los modelos ARIMA y del 98% con respecto al algoritmo original de GP [12].

- **Aproximación serie LYNX**

  La serie LYNX corresponde al número de linces canadienses atrapados por año en el distrito del río Mckenzie del norte de Canadá entre los años 1821 y 1934 (114 observaciones). En la literatura se suele modelar la serie transformándola por medio del logaritmo base 10 y tomando los primeros 100 datos para el entrenamiento y los 14 restantes para la predicción. Martínez [12] presenta los resultados obtenidos al aplicar el algoritmo gpTool, los cuales, al ser comparados con los resultados obtenidos en la literatura con modelos ARIMA, MLP, SVM y el modelo original de GP, se puede concluir que el modelo generado por el algoritmo gpModel posee un SSE menor a los reportados por los demás modelos, con una diferencia del 84% con respecto a los modelos ARIMA y 87% con respecto al algoritmo original de GP. Si bien los valores de SSE de gpTool son similares al de los modelos MLP y SVM, el valor de MAD es menor para gpTool, mostrando menor aleatoriedad [12].

- **Aproximación serie POLLUTION**

  La serie POLLUTION contiene los datos de la cantidad de despachos mensuales de un equipo de polución en miles de francos franceses entre enero de 1986 (ENE1986) y octubre de 1996 (OCT1996), con un total de 130 observaciones. La serie a ser pronosticada corresponde al logaritmo natural de los datos originales; las primeras 106 observaciones son usadas para la estimación de los modelos, mientras que las 24 restantes son usadas para su pronóstico [16, 17]. Los resultados presentados por el algoritmo gpTool propuesto en [12], comparados con los resultados obtenidos en la literatura con modelos ARIMA, MLP, SVM, ANN y el modelo original de GP, muestran que el gpModel posee un promedio de errores cuadradicos (MSE, por su sigla en inglés, equivalente al promedio de las diferencias cuadráticas entre los resultados generados por el modelo propuesto y los valores reales registrados por la serie de tiempo) menor a los reportados por los demás modelos, con una diferencia en la predicción a dos años del 10% con respecto a los modelos SVM, 10% con respecto a las ANN, 10% con respecto a los modelos MPL, 19% con respecto a los modelos ARIMA y 74% con respecto al algoritmo original de GP [12].

- **Aproximación serie INTERNET**

  La serie INTERNET corresponde a la cantidad de usuarios que acceden a un servidor de internet por minuto durante 100 minutos consecutivos. Los primeros 80 datos son usados para la estimación del modelo, y los 20 restantes para su predicción. En la literatura se suele modelar la serie original sin ningún tipo de transformación [16]. Los resultados presentados por el algoritmo gpTool, comparados con los resultados obtenidos en la literatura con modelos ARIMA, MLP, SVM y el modelo original de GP, muestran que el gpModel posee un MSE menor a los reportados por los demás modelos, con una diferencia del 56% con respecto a los modelos SVM,

41% con respecto a los modelos MPL, 58% con respecto a los modelos ARIMA y 61% con respecto al algoritmo original de GP [12].

- **Aproximación serie SUNSPOT**
  La serie SUNSPOT corresponde a la cantidad anual de manchas solares observadas a partir del año 1700; siendo empleada para medir la efectividad de aproximación de estadísticos no lineales. En la literatura se ha modelado la serie original sin ningún tipo de transformación, tomando los primeros 221 datos para el entrenamiento y los 35 restantes para la predicción. Martínez [12] presenta los resultados obtenidos al aplicar el algoritmo gpTool, que al compararse con los resultados obtenidos en la literatura con modelos ARIMA, NN, DAN2, hibrido y el modelo original de GP, se aprecia que el gpModel posee un MSE menor a los reportados por los demás modelos, con una diferencia del 55% con respecto al modelo híbrido, 61% con respecto a los modelos ARIMA, 59% con respecto a los modelos de NN, 41% con respecto a los modelos DAN2 y 96% con respecto al algoritmo original de GP [12].

1.3 Revisión la Literatura

En esta sección se realiza un análisis de la literatura actual en torno a la programación genética, enfocada principalmente a los cambios estructurales del algoritmo y su aplicación a la predicción de series de tiempo. Este análisis complementa el realizado preliminarmente por Martinez [12] en su tesis de maestría, para lo cual se utilizó la metodología de Revisión Sistemática de la Literatura (SLR, por su sigla en inglés) propuesta por Kitchenham et al. [18], ampliamente utilizada en el análisis de evidencia científica.

Para el desarrollo de la metodología de revisión de la literatura empleada, esta sección se estructura de la siguiente manera: en la Subsección 1.3.1 se describe la metodología empleada; en la Subección 1.3.2 se plantean las preguntas de investigación; en la Subsección 1.3.3 se analiza la metodología de la investigación; en la Subsección 1.3.4 se definen los criterios de inclusión y exclusión; en la Subsección 1.3.5 se definen los criterios de calidad a emplearse; en la Subsección 1.3.6 se muestra el proceso de recolección de datos; en la Subsección 1.3.7 se realiza el análisis de los datos; en la Subsección 1.3.8 se muestran los resultados obtenidos; en la Subsección 1.3.9 se realiza la evaluación de calidad de acuerdo con los criterios definidos; en la Subsección 1.3.10 se analizan los factores de calidad; en la Subsección 1.3.11 se realiza la discusión a partir de los resultados de la búsqueda; finalmente, en la Subsección 1.3.12 se muestran las conclusiones de la revisión sistematica de la literatura en torno a la programación genética.

1.3.1 Metodología empleada

La metodología empleada en este estudio se basa en la guía original de la Revisión Sistemática de la Literatura, la cual surge a partir de los estudios realizados por Kitchenham et al. [18], quienes sugieren que la metodología SLR es adecuada como plataforma de análisis de evidencia científica, y específicamente para ingeniería de software. La SLR consta de los pasos: buscar, evaluar, clasificar y analizar. Al aplicar la SRL se buscar analizar la contribución y pertinencia de los distintos artículos de una manera clara, estructurada y reproducible de acuerdo a unos criterios de búsqueda definidos [18-20].

Por lo anterior, para el análisis de las distintas publicaciones en torno a la programación genética y la regresión simbólica (caso particular de GP), se utilizó la metodología SLR con una revisión y catalogación manual a partir de fuentes de búsqueda definidas.

1.3.2 Preguntas de investigación

El objetivo de esta subsección es definir las preguntas de investigación a ser empleadas en la metodología de SLR para el análisis de los principales avances en la literatura de programación genética.

Las preguntas de investigación a ser empleadas son:

- *¿Cuál ha sido la producción investigativa en torno a la programación genética en predicción de series de tiempo?*
  La motivación de esta pregunta es determinar la importancia de emplear la programación genética en la predicción de series de tiempo, identificando la cantidad de estudios publicados año a año, los autores con mayor número de publicaciones, los artículos más citados, y las principales fuentes de publicación.

- *¿Cuáles han sido las hibridaciones propuestas entre el algoritmo de programación genética y otras técnicas estadísticas y de inteligencia computacional para la predicción de series de tiempo?*
  El propósito de la pregunta es identificar aquellas hibridaciones propuestas entre el algoritmo de GP y otras técnicas, que permitan generar resultados más competitivos en la predicción de series de tiempo, analizando la estructura propuesta y las condiciones propias de aplicación.

- *¿Cuáles han sido los cambios propuestos al algoritmo original de programación genética para la predicción de series de tiempo?*
  Esta pregunta surge de la necesidad de conocer aquellos cambios en estructura, pasos y método de búsqueda empleados en la aplicación del algoritmo de GP a la predicción de series de tiempo, con el fin de identificar implementaciones más competitivas de GP que puedan generar mejores modelos en términos del error de predicción.

- *¿Cuáles son los principales hallazgos en la aplicación de la programación genética a la predicción de series de tiempo?*
  Esta pregunta pretende encontrar aquellos puntos de interés alrededor de la aplicación de la GP a la predicción de series de tiempo, identificando oportunidades y posibles falencias no trabajadas; además, condiciones específicas de análisis a considerar para su correcto desarrollo.

- *¿Cuáles de los cambios propuestos al algoritmo original de programación genética han sido utilizados en la predicción de series de tiempo con resultados de error de predicción menores que los reportados por los benchmark de la literatura?*
  La motivación de esta pregunta es identificar cuáles de las modificaciones propuestas al algoritmo original de GP generan un menor error de predicción que los modelos tradicionales de predicción de series de tiempo reportados en la literatura, y bajo qué condiciones (parámetros, tipo de serie de tiempo, entre otros) tienen dicho comportamiento.

1.3.3 Metodología de la investigación

El proceso de búsqueda de los artículos fue realizado de manera automática con refinamiento manual a partir de las fuentes de datos incluidas en SCOPUS (libros, revistas especializadas, artículos y conferencias de GP), para el el periodo de tiempo comprendido entre 1992 y 2016, empleando para ello la siguiente cadena de búsqueda como criterio de selección:

TITLE-ABS-KEY ((genetic programming OR symbolic regression) AND (hybrid OR change OR new OR analysis OR review OR time series OR forecasting OR model OR regression OR method)) AND (PUBYEAR > 1991 AND PUBYEAR < 2017).

Desarrollando los siguientes pasos:

- Búsqueda de los artículos por palabra clave en las fuentes de datos.
- Preselección de los artículos a partir de la verificación de los títulos y fuente de publicación.
- Lectura del resumen (abstract) e identificación del tipo de uso y aplicación de la GP en el artículo, clasificándolos en:
    - Metodología: Cambios en la estructura del algoritmo original o hibridación con alguna técnica existente.
    - Aplicación: Aplicación del algoritmo de GP a un caso específico.
- Exclusión de los artículos que cumplen con al menos un criterio de exclusión.
- Lectura de los artículos que de acuerdo a su resumen puedan dar respuesta a las preguntas de investigación.
- Validación de la relevancia del mismo a partir de los criterios de evaluación de la calidad.

1.3.4 Criterios de inclusión y exclusión

Se incluyeron todos los artículos que cumplieran alguno de los siguientes criterios de inclusión:

- El estudio presenta un cambio en los pasos del algoritmo original de GP o una hibridación con otras técnicas que sugieran cambios en la estructura del mismo.
- El estudio incluye el análisis o predicción de, al menos, una serie de tiempo.

Se excluyen aquellos artículos que cumplieran alguno de los siguientes criterios de exclusión:

- Se presenta una aplicación de GP a un caso particular sin un desarrollo teórico formal.
- En el trabajo analizado no se define una pregunta de investigación, principalmente aquellas referentes a cambios en los pasos, estructura y componentes del algoritmo original de GP.
- Los cambios propuestos se limitan a cambios en las técnicas utilizadas en la optimización de parámetros o valores específicos en los mismos, más no en la inclusión de nuevos, ni hibridación con otras técnicas.
- No presenta resultados comparativos con otras técnicas de la literatura de predicción de series de tiempo, en la predicción de las series de tiempo utilizadas.

1.3.5 Evaluación de la calidad

Para evaluar la calidad de los artículos seleccionados se utilizaron las siguientes preguntas (QQ, por su sigla en inglés):

QQ1. ¿Se realiza una modelación del cambio propuesto en el algoritmo de GP o hibridación?
QQ2. ¿Son analizadas las bondades del cambio propuesto o hibridación en el algoritmo de GP?
QQ3. ¿El estudio presenta resultados comparativos en su aplicación a la predicción de series de tiempo?
QQ4. ¿Se muestra algún procedimiento para la selección apropiada de parámetros del algoritmo?

Se dieron puntajes a las preguntas de la siguiente forma:

QQ1: S (si), la modelación matemática del cambio propuesto en el algoritmo es presentada de forma explícita en el artículo; P (parcialmente), la formulación es implícita; N (no), no se define explícitamente la formulación ni es posible deducirla fácilmente del artículo.

QQ2: S (si), en el artículo son explicadas las implicaciones de la introducción del cambio propuesto en el algoritmo de GP de manera explícita; P (parcialmente), se explica de manera implícita las implicaciones de los cambios propuestos; N (no), no se define explícitamente las implicaciones de los cambios propuestos al algoritmo de GP, ni es posible deducirlos fácilmente del artículo.

QQ3: S (si), el artículo presenta resultados de la aplicación de la solución propuesta a series benchmark de predicción de series de tiempo; P (parcialmente), la formulación muestra resultados de la aplicación y análisis a funciones específicas que no corresponden a series benchmark de predicción de series de tiempo; N (no), no muestra resultados de la aplicación de las solución propuesta.

QQ4: S (si), se realiza un desarrollo conceptual del proceso de búsqueda de manera explícita en el artículo; P (parcialmente), el desarrollo conceptual del proceso de búsqueda es implícito; N (no), no se define explícitamente el método de búsqueda, ni es posible deducirla fácilmente del artículo.

Para las preguntas QQ1 a QQ4 se asignaron los puntajes: $S = 1$, $P = 0{,}5$ y $N = 0$. Como complemento al análisis de los artículos seleccionados, se consideraron los siguientes criterios:

QQ5: Valor del indicador "*Scimago Journal Rank*" (valor de las citaciones ponderada por documento de acuerdo al año de publicación del artículo), el cual refleja el prestigio de la fuente.

QQ6: Valor del indicador "*Source-Normalized Impact per Paper*" (SNIP, corresponde a la razón del conteo de citaciones de la revista por artículo y el potencial de citaciones en el campo de interés).

Adicionalmente, fue analizado el tipo de cambio en la estructura del modelo: Cambio en los pasos del algoritmo, cambio en la determinación de parámetros, hibridación con modelos estadísticos, hibridación con modelos de inteligencia computacional.

1.3.6 Recolección de datos

Para cada uno de los estudios seleccionados fue extraída la siguiente información:

- La fuente del estudio (libro, conferencia, revista) y sus referencias.
- Título y tema principal.
- Autores e institución a la cual pertenecen.
- Resumen (abstract).
- Contenido general.
- Identificación de la pregunta de investigación que resuelve.

Si bien se recopilo toda la información señalada, la evaluación de la calidad y lectura del contenido general, solo fue realizada a aquellos artículos, que de acuerdo con las demás elementos (fuente, titulo y resumen), pudieran responder a las preguntas de investigación de una manera adecuada.

1.3.7 Análisis de datos

Los datos fueron tabulados para responder en orden a las preguntas de investigación de la siguiente manera:

- Análisis especifico de los estudios encontrados en los que se muestre cambios en los pasos del algoritmo original de GP y sus respectivas hibridaciones.
- Análisis de los principales problemas persistentes en la aplicación de la GP a la predicción de series de tiempo, mostrando el estado actual de investigación en el mismo.
- Análisis especifico de los principales avances conceptuales en la aplicación de la GP a la predicción de series de tiempo.
- Análisis y discusión de las principales direcciones de investigación en torno a GP, e identificación de aquellas investigaciones futuras planteadas.

1.3.8 Resultados obtenidos

Al usar la cadena de búsqueda (descrita en la Subsección 1.3.3) en el sistema SCOPUS se recuperó automáticamente un total de 25.431 artículos, de los cuales se aplicó los criterios de inclusión y exclusión de manera automática generando un total de 719 publicaciones. A cada una de las publicaciones se les aplico los criterios de inclusión y exclusión de forma manual, seleccionando un total de 66 artículos, cuyas referencias aparecen en la Tabla 1-1.

Cada uno de los artículos seleccionados fueron catalogados de acuerdo al tipo de aporte (columna 4 de la Tabla 1-1) en: H: Hibridación con alguna técnica, I: Cambio estructural en los individuos, S: Cambio en la selección de los individuos, F: Cambio en la función de aptitud (fitness), M: Cambio en el operador de mutación, C: Cambio en el operador de cruce, A: Cambio general en el algoritmo.

Se resalta que en la Tabla 1-1 gran parte de los artículos seleccionados corresponden a hibridaciones (tipo aporte: H) con otras técnicas de inteligencia computacional (34%), mientras que cambios en los operadores genéticos corresponden a solo el 12% (tipo de aporte: M o C) y aquellos en los que involucra un cambio general en el algoritmo (tipo de aporte: A) el 6%.

Adicionalmente, se puede apreciar un crecimiento constante en el número de publicaciones que cumplen los criterios de selección (conteo de artículos de la Tabla 1.1) a partir del año 2003, con un máximo de 10 artículos/año en los años 2010 y 2012.

1.3.9 Evaluación de la calidad

Para los 66 estudios finalmente seleccionados, se aplicaron los criterios de calidad definidos en la Subsección 1.3.5. Los resultados obtenidos son presentados en la Tabla 1-1, en la que cada fila representa uno de los estudios seleccionados. En la columna 3 son presentados los aportes de los trabajos; las columnas 5 a 8 muestran el grado de cumplimento de los criterios de calidad definidos por las preguntas QQ1 a QQ4; la columna 9 recopila el puntaje total obtenido en relación a las preguntas QQ1 a QQ4; las columnas 10 y 11 muestran los resultados obtenidos al aplicar los criterios complementarios QQ5 y QQ6.

Los resultados del análisis de calidad muestran que el puntaje promedio es $2{,}63 \pm 0{,}34$, lo que significa que la calidad promedio de los estudios seleccionados varía entre $2{,}29$ y $2{,}97$. Nótese que este rango de valores difiere significativamente del puntaje ideal de 4, el cual representa un artículo que cumple con todos los requerimientos necesarios para tener una estrategia apropiada para la selección del modelo final. Esto indica que, en general, los modelos desarrollados entre el 1992 y el 2016 no satisfacen completamente todos los requisitos para desarrollar una metodología de generación de modelos de predicción de series de tiempo basada en el algoritmo de programación genética.

Finalmente, debe señalarse que, aunque todos los artículos seleccionados cumplen con los QQ1 y QQ2, únicamente los estudios [11, 21-23] cumplen parcialmente con el QQ4, referente a la especificación de los parámetros necesarios de ejecución del algoritmo.

Por otro lado, cabe resaltar que si bien la tesis de maestría [12] no fue incluida en la Tabla 1-1 debido a que no esta indexada en SCOPUS, posee una calificación de $3{,}5$ con un desarrollo conceptual parcial del proceso de búsqueda.

1.3.10 Factores de calidad

Cuando se analizaron los 66 artículos seleccionados, se encontró que la mayoría se enfocan en la hibridación con otras técnicas (H), métodos de control del tamaño (F), y selección de los individuos (S); tan solo 16 artículos involucran cambios en la estructura del algoritmo (A) o los individuos de GP (I).

Por otra parte, fue analizada la correlación entre el puntaje obtenido y el tipo de estudio. La Figura 1-4 muestra que los estudios de los tipos de aporte H y S, A y F, y C e I, están caracterizados por valores promedio similares, mientras que los artículos de tipo de aporte M no muestran variabilidad debido a que solo registra un estudio. Nótese que los estudios están ubicados entre los puntajes totales 2,5 y 3 con valores mínimos de 2; además, no existen artículos con calificación de 4.

Al analizar los indicadores SJR y SNIP (columnas 10 y 11 de la Tabla 1-1), se encontró que no hay una relación significativa entre ellos y el puntaje total obtenido. El coeficiente de correlación de Spearman entre el SJR y el puntaje fue 0,203 ($valor - p = 0{,}319$) y entre el SNIP y el puntaje fue 0,057 ($valor - p = 0{,}774$).

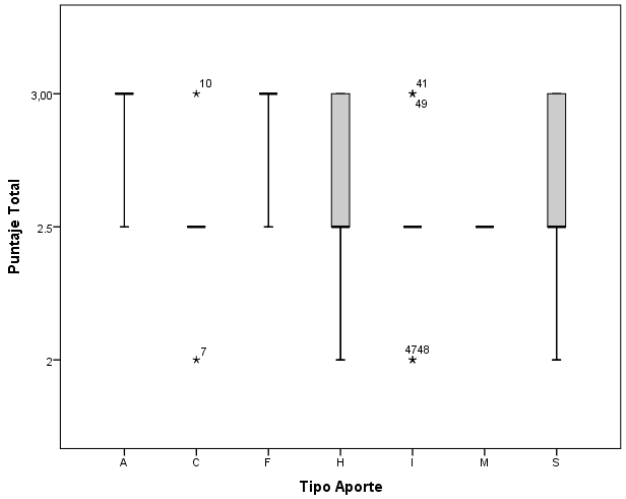


Figura 1-4: Diagrama de cajas de los puntajes de calidad de los estudios seleccionados por tipo de aporte.

Tabla 1-1: Estudios seleccionados.

Ref	Autor /Año	Aporte	Tipo Aporte	QQ1	QQ2	QQ3	QQ4	Puntaje Total	QQ5	QQ6
[24]	McPhee & Poli (2008)	Se propone el algoritmo de memoria con memoria (MwM-GP) en el que se mantienen algunos individuos entre generaciones a partir del uso de operadores de asignación y retorno suave.	A	S	S	S	N	3		
[25]	Xie et al. (2007)	Se propone disminuir la terminación prematura del proceso de búsqueda por medio de la agregación de información adicional a los individuos de GEP y su hibridación con SA.	A	S	S	P	N	2,5	0,269	0,373
[8]	Hoang et al. (2006)	Se propone la generación de individuos de GP por medio de TAG, dividiendo el conjunto de terminales entre no terminales y terminales.	A	S	S	S	N	3		
[26]	Xiong et al. (2003)	Se propone el algoritmo PTGP, basado en GP, el cual agrega un vector de discontinuidades para separar los individuos que aportan a cada una de ellas.	A	S	S	S	N	3	0,141	0,196

Tabla 1-1: Estudios seleccionados (Continuación).

Ref	Autor /Año	Aporte	Tipo Aporte	QQ1	QQ2	QQ3	QQ4	Puntaje Total	QQ5	QQ6
[27]	Ono et al. (2012)	Se propone un nuevo operador de cruce en el que cada nodo de los individuos posee una probabilidad de aplicación del operador genético de acuerdo a su posición en el árbol.	C	S	S	P	N	2,5		
[28]	Hanada et al. (2012)	Se propone el uso del método de MSXF y dMSXF como operador de cruce para GP. dMSXF como un caso particular de MSXF a partir de una regla determinística basada en la medida de distancia.	C	S	S	P	N	2,5		
[29]	Semenkin & Semenki-na (2012)	Se propone un nuevo operador de cruce para el algoritmo de GEP basado en la probabilidad de selección de los genes a cruzar a partir de su valor de aptitud.	C	S	S	N	N	2		
[30]	Uy et al. (2010)	Se propone el operador genético SSC para el algoritmo de GP basado en SAC con un control semántico.	C	S	S	P	N	2,5	0,331	0,545
[31]	Vanneschi & Gustafson (2009)	Se propone el uso de dos capas de selección de los individuos y una memoria de exclusión de individuos llamados repulsores durante la fase de búsqueda de GP.	C	S	S	P	N	2,5		
[21]	Vafaee et al. (2008)	Se propone un método de adaptación de las probabilidades de aplicación de los operadores genéticos basados en la aptitud general de la población de GP durante la fase de búsqueda.	C	S	S	P	P	3	0,192	0,294
[32]	Terrio et al. (2002)	Se propone un nuevo operador genético de cruce en el que se seleccionan los nodos con mayor valor de aptitud como nodos base para la aplicación del operador tradicional de cruce de GP.	C	S	S	P	N	2,5	0,142	0,270
[33]	Agapitos et al. (2012)	Se propone una nueva medida de aptitud basada en la descomposición de la varianza del error de predicción.	F	S	S	S	N	3		
[34]	Montaña et al. (2011)	Se analiza GP bajo la perspectiva de las máquinas de soporte, comparando las medidas de complejidad contra los métodos basados en la teoría de aprendizaje estadística de Vapnik-Chervonenkis.	F	S	S	S	N	3		
[35]	Imada & Ross (2008)	Se propone una nueva medida de aptitud en la que se considera además de la suma de errores, otras medidas estadísticas adicionales.	F	S	S	S	N	3		
[36]	Zhang & Nandi (2007)	Se proponen dos operadores de control de descendencia durante el proceso de búsqueda del algoritmo de GP para la disminución del tamaño de los individuos de GP.	F	S	S	P	N	2,5	2,191	3,509
[37]	Xie et al. (2006)	Se propone generar subconjuntos de individuos con características similares (clúster) y evaluar la aptitud de solo un individuo representante por clúster.	F	S	S	P	N	2,5		
[38]	Tomassini et al. (2004)	Se propone una metodología de eliminación de individuos irrelevantes de la población basada en su valor de aptitud y una tasa variable de selección entre generaciones.	F	S	S	S	N	3		1,198

Tabla 1-1: Estudios seleccionados (Continuación).

Ref	Autor /Año	Aporte	Tipo Aporte	QQ1	QQ2	QQ3	QQ4	Puntaje Total	QQ5	QQ6
[39]	Sánchez (2000)	Se propone un método de generación de intervalos de confianza para los individuos de la población de GP.	F	S	S	S	N	3	4,226	5,781
[40]	Karaboga et al. (2012)	Se propone la hibridación entre los algoritmos de ABC y GP llamado ABCP, reemplazando las abejas obreras de ABC por los individuos de GP.	H	S	S	S	N	3	2,393	3,001
[41]	Chan & Kwong (2012)	Se propone la hibridación de la regresión Fuzzy con GP llamada HFR, en la que se toma como base el algoritmo de GP y se modifican los parámetros de los individuos de GP por parámetros Fuzzy.	H	S	S	N	N	2		
[42]	Kronber-ger & Beham (2012)	Se propone la hibridación entre los algoritmos de búsqueda tabú y GP, en la que se introduce una memoria de largo plazo de validación de la forma semántica de los individuos de GP, para ser descartarlos de la población entre generaciones.	H	S	S	P	N	2,5		
[43]	Guo et al. (2010)	Se propone la hibridación entre los algoritmos de EM y GP llamado GP-EM.	H	S	S	N	N	2		
[44]	Borges et al. (2010)	Se propone el uso de EE y GP con un desarrollo de SLP, en el que los individuos de SLP son reemplazados por los de GP y optimizados por medio de EE.	H	S	S	P	N	2,5		
[45]	Gandomi et al. (2010)	Se propone la hibridación de los algoritmos OLS y GP llamado GP/OLS, en el que se realiza una poda sucesiva de los subárboles de los individuos de GP por medio del cálculo del aporte de los mismos a la predicción total del individuo por medio de OLS.	H	S	S	S	N	3	0,449	0,778
[46]	Pennachin et al. (2010)	Se propone la hibridación entre GP y AA llamado AA+, en el que se analizan los rangos de los terminales para descartar aquellos por fuera del rango de los datos de entrenamiento.	H	S	S	P	N	2,5		
[47]	Yuen & Leung (2009)	Se propone el algoritmo NsGP, en el que se almacenan las soluciones de GP en formato S-expression para no ser utilizadas nuevamente (similar a búsqueda tabú).	H	S	S	P	N	2,5		
[48]	Barton et al. (2009)	Se propone el algoritmo NN-GP, en el que se utiliza una NN y se genera la función de transición de cada neurona por medio de GP.	H	S	S	P	N	2,5	0,939	1,825
[49]	Fillon & Bartoli (2007)	Se propone el algoritmo GP-HVES, el cual parte los datos de entrada por medio de HVES generando modelos de GP para cada discontinuidad (grupo de datos de entrada).	H	S	S	P	N	2,5		
[50]	Giustolisi & Savic (2006)	Se presenta el algoritmo EPR, el cual es la hibridación de GP y la regresión numérica, en la que se penalizan los individuos de GP por su grado de complejidad.	H	S	S	P	N	2,5	0,424	1,081

Tabla 1-1: Estudios seleccionados (Continuación).

Ref	Autor /Año	Aporte	Tipo Aporte	QQ1	QQ2	QQ3	QQ4	Puntaje Total	QQ5	QQ6
[51]	De Menezes & Nikolaev (2006)	Se presenta una hibridación entre NN y GP, en la cual se reformulan las neuronas de NN como bloques polinomiales y se calcula la aptitud del individuo de GP como una suma ponderada de la aptitud de sus subárboles.	H	S	S	S	N	3	1,363	1,566
[52]	Smits & Vladislav-leva (2006)	Se propone la hibridación de los algoritmos de GP y OP llamado OrdinalParetoGP, en el que se optimizan los individuos de GP por medio de OP.	H	S	S	S	N	3		
[53]	Costa & Pozo (2006)	Se propone la hibridación de GP con EE, en el que se cambia el método de selección de los individuos para la aplicación de los operadores genéticos por la metodología $\mu + \lambda$.	H	S	S	P	N	2,5		
[54]	Cagnoni et al. (2005)	Se propone la hibridación entre los algoritmos de GP y GA llamado GGPA, en el que se optimizan los terminales de GP por medio de GA.	H	S	S	P	N	2,5		
[55]	Jiang et al. (2005)	Se propone la hibridación entre GEP y SA, introduciendo el concepto de temperatura a cada individuo como la probabilidad de seleccionarlo para la aplicación del operador genético de mutación.	H	S	S	S	N	3		
[56]	Jiang et al. (2005)	Se propone la hibridación entre GEP y SA, adicionandole a los individuos de GP la probabilidad de selección para la aplicación del operador genético de mutación como una reinterpretación de la temperatura de SA.	H	S	S	S	N	3	0,384	0,892
[57]	Oltean (2004)	Se propone la hibridación entre los algoritmos de GP y TGP, en el que se modifica el operador de cruce y se introduce el operador de inserción (inserta a cada individuo de la población un subárbol especifico).	H	S	S	P	N	2,5		1,198
[58]	Lee (2001)	Se propone la hibridación entre los algoritmos de TSPA y GP, en el que se toma como base el algoritmo de TSPA identificando las funciones a ser utilizadas por medio de GP.	H	S	S	S	N	3		
[59]	Howard et al. (1995)	Se propone la hibridación entre los algoritmos de GP y GA llamado GA-P, en el que se optimizan los individuos de GP por medio de GA.	H	S	S	S	N	3		
[60]	Zăvoianu et al. (2012)	Se propone el uso de DDL e IPT para la reducción del tamaño de los individuos de GP, manteniendo similar el error de predicción de los mismos.	I	S	S	P	P	3	0,331	0,545
[61]	Tanji & Iba (2010)	Se propone el algoritmo PORTS, en el que se modifican los individuos y el operador genético de cruce, para que sea posible tomar más de dos individuos padre para la generación de los individuos hijo.	I	S	S	P	N	2,5	0,231	0,446

Tabla 1-1: Estudios seleccionados (Continuación).

Ref	Autor /Año	Aporte	Tipo Aporte	QQ1	QQ2	QQ3	QQ4	Puntaje Total	QQ5	QQ6
[10]	Johnston et al. (2010)	Se propone un método de simplificación de nodos redundantes en los individuos de GP por medio de los métodos de simplificación de rango y la remoción de hijos redundantes; además, de la inclusión del operador de simplificación de regresión lineal.	I	S	S	P	N	2,5	0,331	0,545
[62]	Mori et al. (2009)	Se presenta el método EDS para la simplificación de los individuos de GP por medio de la evaluación de la función de aptitud en cada uno de los subárboles del individuo.	I	S	S	P	N	2,5	0,331	0,545
[63]	Kinzett et al. (2009)	Se propone un método de simplificación de los individuos de GP por medio de su codificación en una cadena binaria, reemplazando parte de la misma por un único bit.	I	S	S	P	N	2,5		
[64]	Zhang & Wong (2008)	Se propone la simplificación algebraica de los individuos de GP por medio de una serie de reglas aritméticas. La comparación de los individuos se realiza por medio de métodos hash.	I	S	S	P	N	2,5	0,269	0,533
[65]	Chen et al. (2008)	Se propone el uso de HDN en los individuos de GEP para almacenar el coeficiente y exponente en cada nodo, equivalente a un término de un polinomio.	I	S	S	N	N	2		
[66]	Cerny et al. (2008)	Se propone el uso de DE para la determinación de las constantes a ser utilizadas por los individuos de GP.	I	S	S	N	N	2		
[67]	Muntean et al. (2007)	Se propone el algoritmo de poda de individuos de GP llamado BSTGP, en el que se reemplaza el individuo por el subárbol con mejor valor de aptitud.	I	S	S	S	N	3		
[68]	Li et al. (2006)	Se introduce el concepto de módulos emergentes desacoplados para ser utilizados en la generación de nuevos individuos de GPEP entre generaciones, por medio del operador de compresión.	I	S	S	P	N	2,5		
[7]	Nikolaev & Iba (2001)	Se redefinen los individuos de GP por medio de bloques poligonales y la aplicación del algoritmo de STRAGANOFF.	I	S	S	P	N	2,5		
[11]	Ok et al. (2000)	Se propone un mecanismo de eliminación de terminales irrelevantes en los individuos de GP por medio de la evaluación de su aporte a la aptitud total del individuo.	I	S	S	P	N	2,5	0,331	0,545
[69]	Aichour & Batouche (2009)	Se propone un nuevo operador genético de mutación para GP, en el que se selecciona el subárbol con menor valor de aptitud y se reemplaza por uno nuevo por medio de la aplicación del operador tradicional de mutación de GP.	M	S	S	P	N	2,5	0,197	0,259

Tabla 1-1: Estudios seleccionados (Continuación).

Ref	Autor /Año	Aporte	Tipo Aporte	QQ1	QQ2	QQ3	QQ4	Puntaje Total	QQ5	QQ6
[70]	Tao et al. (2012)	Se propone el uso de DPV durante la fase de búsqueda de GP, redefiniendo la función de escalamiento y pivoteo exponencial de DPV, para la fórmula de determinación de la variación de la población de GP.	S	S	S	S	N	3	0,169	0,417
[23]	Zhao et al. (2012)	Se propone un mecanismo de determinación del tamaño de los datos de entrenamiento basado en HEA.	S	S	S	P	P	3	1,023	2,173
[71]	McDer-mott et al. (2011)	Se presenta una definición formal de distancia entre funciones basada en MSTP, coherente con los operadores genéticos de GP, proveyendo un algoritmo de cálculo 1STP para el operador de mutación de GP.	S	S	S	N	N	2	0,331	0,545
[72]	Kronber-ger et al. (2010)	Se propone un nuevo método de selección de los individuos de GP aplicado inmediatamente después de la aplicación del operador de mutación de GP.	S	S	S	P	N	2,5		
[9]	Borges et al. (2010)	Se propone una nueva medida de complejidad para SRM, aplicada al método de selección de individuos de GP, basada en la teoría de aprendizaje estadística de Vapnik-Chervonenkis.	S	S	S	P	N	2,5		
[73]	McRee (2010)	Se propone el uso de una nueva estructura de datos que almacene los individuos vecinos de cada individuo de la población de GP, con el fin de extender el espacio de búsqueda, concentrándose en las características de cambio entre los mismos.	S	S	S	P	N	2,5		
[74]	Vanneschi & Cuccu (2009)	Se propone el algoritmo DynPopGP para la selección del tamaño de la población de GP en cada generación, a partir del valor de aptitud de cada individuo de la población actual.	S	S	S	S	N	3		
[75]	Wilson & Heywood (2006)	Se propone el algoritmo DynPopGP para la selección del tamaño de la población de GP.	S	S	S	N	N	2		
[76]	Xie (2005)	Se propone la hibridación de GP con EE en el que se utiliza el método $\mu + \lambda$ para la selección de los individuos para la aplicación de los operadores genéticos.	S	S	S	S	N	3	0,331	0,545
[77]	Rochat et al. (2005)	Se propone el mapeo adaptativo durante el proceso de búsqueda de DGP por medio del método Huffman.	S	S	S	P	N	2,5	0,331	0,545
[78]	De Arruda et al. (2014)	Se propone la hibridación entre las SVM y GP, para la extracción de las reglas de clasificación de un grupo de datos.	H	S	S	S	N	3		
[79]	Zamani et al. (2014)	Se propone la hibridación entre KPCA y KLDA, y GP para la combinación de los kernels.	H	S	S	S	N	3		
[80]	Amir et al. (2014)	Se propone la optimización de la estructura funcional de GP por medio de uso de múltiples datos de menor a mayor complejidad.	S	S	S	S	N	3		

1.3.11 Discusión

En esta sección se responden las preguntas de investigación planteadas en la Subsección 1.3.2.

- *¿Cuál ha sido la producción investigativa en torno a la programación genética en predicción de series de tiempo?*

Entre los años 1992 a 2016 se han desarrollado 66 propuestas de cambio en torno al algoritmo de GP en la predicción de series de tiempo, las cuales cumplen todos los criterios de inclusión e exclusión definidos en esta investigación; dichos estudios aparecen relacionados en la Tabla 1-1, discriminados por tipo de aporte. Se resalta que solo el 2% de los trabajos registrados correspondan al tipo de aporte M, mientras que los tipos de aporte H, I y S representan el 34%, 19% y 17% de los estudios respectivamente.

Adicionalmente, de los artículos seleccionados, solo el 31% corresponde a publicaciones en revistas especializadas, mientras que el restante 69% corresponde a artículos de conferencias. Entre las fuentes con mayor número de artículos publicados de la Tabla 1-1, se encuentra la publicación *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* con 11 artículos publicados; y las conferencias: *GECCO'08: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation 2008*, *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics* y *Proceedings of the 11th Annual Genetic and Evolutionary Computation Conference, GECCO-2009*, con 3 artículos publicados cada una de ellas.

- *¿Cuáles han sido las hibridaciones propuestas entre el algoritmo de programación genética y otras técnicas estadísticas y de inteligencia computacional para la predicción de series de tiempo?*

Entre el periodo analizado (1992 a 2016) se presentan un total de 24 propuestas de hibridación con otras técnicas (Tabla 1-1), distribuidas entre dos enfoques de hibridación. El primer enfoque consiste en tomar como base el algoritmo original de GP (o alguna de sus variantes) y utilizar el otro algoritmo para optimizar los parámetros de los individuos; en la Tabla 1-1 se encuentran los trabajos de hibridación con este enfoque entre GP y los algoritmos de: Regresión Fuzzy [41], Búsqueda Tabú [42, 47] (genera memoria de almacenamiento de individuos para ser excluidos en las generaciones subsiguientes, sin modificar el proceso de búsqueda de GP), Algoritmo de maximización de la expectativa (EM) [43], Mínimos cuadrados ortogonales (OSL) [45] (se realizan podas sucesivas de los individuos por medio de OSL), Afinación aritmética (AA) [46] para excluir terminales fuera del rango de los datos, Algoritmo de separación de híper – volúmenes (HVES) [49] para el manejo de discontinuidades, Regresión polinómica evolutiva (EPR) [50] (penaliza los individuos por su grado de complejidad), Redes Neuronales (NN) [51], Optimización ordinal (OP) [52], Algoritmos genéticos (GA) [53] (cambia el método de selección de los individuos por $\mu + \lambda$), GA [54, 59] (optimización de los individuos por medio de GA), Programación genética no ruteada (TGP) [57] (modifica el operador de cruce y se introduce el operador de inserción).

El segundo enfoque se basa en tomar como base el otro algoritmo (no GP) y reemplazar los individuos u objetos de análisis por los individuos de GP; con este enfoque se encuentran en la Tabla 1-1 las siguientes hibridaciones de GP con: el algoritmo de colonia de abejas (ABC) [40], el algoritmo de recocido simulado (SA) [55-56], el algoritmo de programa de línea recta (SLP) [44]; Redes neuronales (NN) [48]; SVM [78]; KPCA y KLDA [79]; y el algoritmo de perturbación de series de tiempo (TSPA) [58].

- *¿Cuáles han sido los cambios propuestos al algoritmo original de programación genética para la predicción de series de tiempo?*

En la literatura analizada y seleccionada en la Tabla 1-1, los cambios presentados al algoritmo de GP fueron catalogados de acuerdo al tipo de aporte. Las hibridaciones con otras técnicas (H) de inteligencia

computacional fueron analizadas en la respuesta a la segunda pregunta de investigación. Los cambios generales en el algoritmo de GP (A) registrados en la Tabla 1-1, corresponden a: la persistencia de componentes de individuos de GP entre generaciones [24]; la agregación de información adicional a los individuos con el fin de evitar la terminación prematura del algoritmo de GP durante la fase de búsqueda de soluciones [25]; la generación de individuos por medio de árboles sintácticos contiguos (TAG) [8]; y, el manejo de discontinuidades [26]. Adicionalmente, se presentan aquellos trabajos enfocados a la concentración de esfuerzos en zonas de interés durante el proceso de búsqueda del algoritmo de GP, entre los que se encuentran: el uso de estructuras de memoria para reutilizar individuos de generaciones anteriores [24, 25]; la generación jerarquica de los individuos [8]; y el manejo de discontinuidades en la serie de tiempo [26].

Los cambios en el operador de cruce (C) reportados por la Tabla 1-1 corresponden a: la inclusión de la probabilidad de aplicación del operador genético de cruce de cada nodo del individuo padre, proporcional a su posición en el individuo [27]; el uso del método de cruce de fusión multipaso (MSXF) como operador genético de cruce [28]; selección de los genes (nodos) a ser cruzados (de los individuos padre) basado en su valor de aptitud [29]; el uso del cruce semántico consciente (SAC) [30]; el uso de dos capas de selección de individuos incluyendo una estructura de memoria de exclusión (almacenamiento de los individuos utilizados para su exclusión en generaciones futuras) [31]; adaptación de la probabilidad de aplicación del operador genético de cruce basado en el valor de aptitud general de la población [21]; y la selección de nodos (de los individuos padre) para la aplicación del operador genético de cruce, basado en el valor de aptitud de su subárbol [32].

Las propuestas de cambio en la función de aptitud (fitness) (F), presentados en la Tabla 1-1, se basan en: la descomposición de la varianza del error de predicción [33]; la consideración de otras medidas estadísticas dentro de la medida de error [35]; la generación de clústers y selección de individuos a partir del calculo de la aptitud de un único individuo por cluster [37]; la eliminación de individuos irrelevantes por medio de una tasa variable de selección [38]; la generación de intervalos de confianza para los individuos de la población de GP [39]; la generación de medidas de complejidad adicionales, analizando GP bajo la perspectiva de las máquinas de soporte [34]; y el control de la descendencia de los individuos durante el proceso de búsqueda del algoritmo de GP [36].

Los cambios estructurales en los individuos (I) presentados en la Tabla 1-1, se enfocan principalmente en la disminución del tamaño de los individuos, para lo cual se propone: el uso del algoritmo de limitación dinámica de profundidad (DDL) [11]; el uso de métodos de simplificación de rango, basados en la remoción de hijos redundantes e inclusión del operador de simplificación de regresión lineal [10]; el uso del método de simplificación de decisión equivalente (EDS) [62]; la codificación binaria de los individuos y posterior reemplazo por bits [63]; la simplificación algebraica a partir de reglas aritméticas [64]; el uso de nodos de alta densidad (HDN) para el almacenamiento de información adicional en los nodos de los individuos de GEP [65]; la evaluación de la función de aptitud de los subárboles internos de los individuos de GP para la generación de la nueva población de hijos [67]; la eliminación de terminales irrelevantes de acuerdo a su aporte a la aptitud total del individuo [11];el uso de evolución diferencial (DE) para la determinación de las constantes a ser utilizadas por los individuos [66]; el uso de módulos emergentes desacoplados persistentes entre generaciones para la generación de nuevos individuos [68]; la redefinición de los individuos de GP por medio de bloques poligonales [7]; y la modificación del operador genético de cruce para ser aplicado en más de dos individuos padre [61].

Por otra parte, la Tabla 1-1 solo registra un cambio en el operador de mutación (M), el cual consiste en la selección del nodo con menor valor de aptitud del individuo, para la posterior aplicación del operador tradicional de mutación de GP sobre el mismo [69].

En la selección de los individuos (S), en la Tabla 1-1 se muestra las propuestas de modificación relacionadas con el cambio del tamaño de la población y métodos de selección de los individuos, los cuales son: el uso del algoritmo de variación dinámica de la población (DPV) para la determinación del tamaño de la población dinámicamente durante el proceso de búsqueda de GP [70]; el cambio de tamaño de la población de acuerdo al valor de aptitud de los individuos que la componen [74, 75]; la aplicación de operadores $\mu + \lambda$ para la selección de los individuos de la población de GP [76]; uso conjunto de los métodos de multipoblación y

selección dinámica adaptativa, para la eliminación de los individuos irrelevantes durante el proceso de búsqueda de GP [77]; la determinación del tamaño adecuado del conjunto de datos de entrenamiento, a partir del uso del índice Hoeffding basado en algoritmos evolutivos (HEA) [23]; cálculo de la distancia entre funciones para la selección de los individuos, basada en la probabilidad de transición multipaso (MSTP) [71]; cambio en el orden de aplicación del paso de selección de los individuos en el algoritmo de GP [72]; cambio en la medida de aptitud empleada, por medio del uso de la minimización del riesgo estructural (SRM), para la selección de los individuos durante el proceso de búsqueda de GP [9]; extensión del espacio de búsqueda, al incluir los vecinos de los individuos como parte de la población de GP [73]; el mapeo adaptativo por medio del método de Huffman [77]; y la optimización de la estructura funcional de los individuos por medio de la evaluación en multiples conjuntos de datos de menor a mayor complejidad [80].

- *¿Cuáles son los principales hallazgos en la aplicación de la programación genética a la predicción de series de tiempo?*

Entre los principales hallazgos encontrados en la aplicación de la GP a la predicción de series de tiempo, se encuentra su versatilidad para la hibridación con otras técnicas de inteligencia computacional [40-59, 78-79], el comportamiento de los individuos ante cambios en la manera de selección de la población de hijos entre generaciones durante el proceso de búsqueda [9, 70-77, 80]; el cambio en los operadores genéticos de cruce [19, 27-32] y mutación [69] basados principalmente en la selección de los nodos de los individuos padre —proporcional a su valor de aptitud— a los cuales se le aplicarán dichos operadores; cambios en la función de aptitud empleada, que incluya información adicional al error de predicción [33-39]; cambios en la estructura de los individuos para la reducción del tamaño excesivo de los individuos [7, 10, 11, 60-68]; y cambios generales en los pasos del algoritmo de GP [8, 24-26]. Cada una de ellas analizada en la Tabla 1-1 y la discusión de la tercera pregunta de investigación.

- *¿Cuáles de los cambios propuestos al algoritmo original de programación genética han sido utilizados en la predicción de series de tiempo con resultados de error de predicción menores que los reportados por los benchmark de la literatura?*

De acuerdo con los trabajos presentados en la Tabla 1-1, y al criterio de calidad QQ3 (columna 7) evaluado, en el cual tiene un S si fue realizada una validación con respecto a series de tiempo benchmark, y una P si no son explícitos o no corresponden con predicción de series de tiempo. Se tiene que solo 20 de los 66 estudios cumplen totalmente con el criterio QQ3, los cuales muestran en casi todos los resultados un mejor comportamiento —en términos del error de predicción— que el algoritmo original de GP.

1.3.12 Conclusiones de la revisión sistemática de la literatura

En esta sección, como primer aporte al trabajo de tesis de doctorado, fueron analizados los artículos publicados en la literatura más relevante durante el periodo 1992 a 2016, en los que se proponen cambios estructurales en el algoritmo original de programación genética para el pronóstico de series de tiempo. Para cada uno de los artículos analizados se discutieron las propuestas de solución a los problemas vigentes que aún persisten en la aplicación de GP. En general, se observó que las propuestas planteadas, en los estudios seleccionados, no satisfacen completamente un procedimiento sistemático de construcción de modelos de predicción de series de tiempo, debido principalmente a la falta de un procedimiento explícito de selección de parámetros de ejecución del algoritmo de GP.

Si bien se han realizado aportes a la modelación de series de tiempo por medio de GP, aun persisten falencias en la inclusión de conocimiento experto en la limitación del espacio de búsqueda de las soluciones, en la alta redundancia de operadores y terminales en la construcción de los individuos, y en la redirección de

los individuos hacia zonas prometedoras de búsqueda por medio de la aplicación de los operadores genéticos de reproducción, cruce y mutación.

A continuación se muestran aquellas falencias que aún persisten en la literatura de programación genética aplicada a la predicción de series de tiempo:

- *Aumento desmedido del tamaño de los individuos de programación genética, sin un aporte adicional a la mejora del error de predicción.* El aumento desmedido del tamaño de los individuos sin aporte a la mejora del valor de aptitud de los mismos, también llamado bloat, es uno de los principales problemas en la aplicación del algoritmo de programación genética, principalmente por sus efectos negativos durante el proceso de búsqueda, entre los que se asocia la degradación de la calidad de los individuos (soluciones) de la población [81, 82].
- *Coexistencia de componentes de la serie de tiempo en la estructura de los individuos de una manera ilógica e incalculable.* Al realizar la interpretación del individuo (función matemática en forma de árbol sintáctico) resultante de la aplicación del algoritmo de GP, con respecto a la realidad del fenómeno, a partir de sus componentes, es importante garantizar las propiedades matemáticas mínimas de los términos empleados. Para el caso de las series de tiempo, cuya función general corresponde a: $F(\mathbf{x}, \boldsymbol{\theta}) : \{[\mathbf{x} \in \mathbb{R}^p, \boldsymbol{\theta} \in \mathbb{R}^m] \rightarrow \mathbb{R} \mid F(\mathbf{x}, \boldsymbol{\theta}) = \lambda_1 S(\mathbf{x}, \boldsymbol{\theta}) + \lambda_2 C(\mathbf{x}, \boldsymbol{\theta}) + \lambda_3 \Gamma(\mathbf{x}, \boldsymbol{\theta}) + \lambda_4 \Upsilon(\mathbf{x}, \boldsymbol{\theta})\}$. Donde, S es la componente de estacionalidad, C es la componente de ciclo, Γ es la componente de tendencia, Υ es la componente de error, y $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$ son coeficientes dicotómicos [0,1] que indican si aplica o no la componente de acuerdo con restricciones matemáticas. Es importante garantizar durante el proceso de búsqueda que los modelos generados cumplan con las características de un modelo general de series temporales, en estructura y restricciones.
- *Falta de focalización y limitación del espacio de búsqueda en zonas prometedoras*. Dado que el espacio de búsqueda del algoritmo de GP es infinito, formado por todas las combinaciones de operadores y terminales definidos ($\{Z \circledast T\}$, donde el operador $\circledast$ genera todas las posibles combinaciones entre los elementos de los dos vectores Z y T tal que la combinación resultante es de tamaño infinito al permitirse el uso de los elementos de cada vector más de una vez, y las combinaciones entre elementos $T_{1 \leq i \leq l}$ del vector T a través de al menos un elemento $\zeta_{1 \leq i \leq k}$ del vector Z; lo que implica que, no es posible combinaciones de solo elementos consecutivos del vector Z ($\{Z \circledast T\} \cup \{\zeta_1, \dots, \zeta_l, \dots, \zeta_i \dots \zeta_j\} = \emptyset$), por lo que $\{Z \circledast T\} = \{T_1, T_2, \dots, T_l, T_1\zeta_1 T_1, T_1\zeta_1 T_2, \dots, T_2\zeta_1 T_1, \dots, T_1\zeta_2 T_1, \dots, T_1\zeta_1 T_1\zeta_2 T_1, \dots, T_1\zeta_1 T_2\zeta_2 T_1, \dots, T_i\zeta_j \dots T_a, \dots\}$).

  Por lo anterior, la limitación del espacio de búsqueda y la focalización en zonas prometedoras cobra gran relevancia a nivel estructural (individuos de menor tamaño y con mejor medida de aptitud) y en tiempo computacional empleado. La focalización del espacio de búsqueda permite centrarse en aquellas estructuras funcionales más prometedoras (subconjunto de los elementos de $\{Z \circledast T\}$), generando individuos, que en teoría, generan un valor de aptitud superior al de otras zonas.
- *Falta de inclusión de los principales modelos de predicción de series de tiempo durante la generación de los individuos.* Los modelos de predicción de series de tiempo presentes en la literatura, corresponden a la base teórica actual del desarrollo de nuevos modelos, hibridación de los existentes, y la modificación de los actuales. Además, el objetivo del desarrollo de nuevos modelos de predicción es la superación de falencias en los modelos actuales, es necesario garantizar el uso de los mismos durante el proceso de búsqueda del algoritmo de GP.
- *Operadores genéticos inconsistentes con la definición de algoritmos genéticos*. El proceso de búsqueda del algoritmo de GP se basa en la aplicación de los operadores genéticos. Los operadores genéticos de GP corresponden a una extensión de los operadores genéticos de los algoritmos genéticos, los cuales tienen el objetivo de introducir pequeños cambios (mutación), y

mezcla de aquellas componentes (cruce) que redireccionen a los individuos en el sentido de la mejora del valor de la aptitud de los mismos (suele ser la disminución del error de predicción); sin embargo, al analizar la definición de los operadores actuales de mutación y cruce aplicados al algoritmo de GP, no son consistentes con la definición original, debido principalmente, a la imposibilidad de sugerir cambios en la estructura de los individuos en la dirección del mínimo local más cercano en el espacio de búsqueda definido, lo que conlleva a que los individuos recorran de una manera meramente aleatoria el espacio de búsqueda, con los costos computacionales y de no mejora de la función de aptitud asociados.

- *Falta de inclusión de todos los rezagos relevantes en cada individuo de la población durante el proceso de búsqueda del algoritmo de programación genética.* Los rezagos constituyen las variables de entrada de las series temporales. En el algoritmo de programación genética, los rezagos son incluidos en el conjunto de terminales, al igual que las constantes definidas. Debido a que los terminales son seleccionados de manera aleatoria para la generación de los individuos durante el proceso de búsqueda del algoritmo de GP, no existe garantía de uso de todos los rezagos definidos, por lo que no es posible, en muchos casos, lograr la adecuada aproximación de la serie de tiempo por medio del modelo arrojado por GP, e inclusive, asociarle un mejor valor de aptitud a los individuos con ciertos terminales específicos, y no por tener una forma funcional superior; lo que implica que un individuo I_1 puede ser mejor que un individuo I_2 debido a que se incluye algún rezago de mayor influencia en la salida, y no porque posea una estructura funcional superior, desvirtuándose el sentido del proceso de búsqueda de GP.
- *Selección inadecuada de las constantes a ser incluidas en el conjunto de terminales.* La definición de las constantes a ser incluidas en el conjunto de terminales, es un problema recurrente al momento de utilizar el algoritmo de GP, debido a su influencia en el nivel de precisión de los parámetros asociados a los rezagos, y por ende, al valor de aptitud de los individuos. Si bien se han propuesto el uso de evolución incremental para la determinación de las constantes a ser utilizadas por los individuos de GP [60], y un mecanismo de eliminación de terminales irrelevantes en los individuos de GP, por medio de la evaluación de su aporte a la aptitud total del individuo [64]; aún persisten ineficiencias en el proceso de búsqueda asociados a la selección y optimización de las constantes que afecta directamente la calidad de la solución encontrada.

  Debido a que las constantes constituyen los parámetros optimizados de los individuos durante el proceso de búsqueda tradicional de GP; la cantidad de constantes, sus valores, el nivel de precisión y el método de adaptación (cuando corresponden a constantes dinámicas) constituye una línea de investigación activa en la aplicación del algoritmo de GP a la predicción de series temporales, como se puede apreciar en la Tabla 1-1. Además, constituyen una parte fundamental de los terminales, la correcta selección de los mismos constituye uno de los principales aspectos a considerar durante la aplicación de GP, debido a que una definición inadecuada de ellos puede conllevar a que los individuos deambulen por el espacio de búsqueda sin una dirección clara.

1.4 Preguntas emergentes

Desde la introducción de la GP por Koza [3] en 1992, se han generado numerosos artículos mostrando las ventajas de la aplicación del algoritmo de GP a problemas específicos, algunas hibridaciones con otras técnicas estadísticas y de inteligencia computacional, y otros enfocados a solventar los problemas relacionados con la aplicación de dicha técnica.

Algunos de los principales problemas que se han abordado en la literatura actual de GP han sido: la redundancia en los operadores de los modelos resultantes (generación ecuaciones/individuos muy extensos que no aportan a la mejora de la aptitud), la falta de verificación de la compatibilidad lógica-matemática entre componentes (terminales y operadores) de los individuos resultantes, y la falta de jerarquía en la generación

de los individuos (estructura redundante de terminales y operadores sin una equivalencia clara a funciones de predicción de series de tiempo presentes en la literatura). Dichos problemas han generado vacíos y necesidades crecientes de mejora, en la aplicación del algoritmo de GP a la predicción de series de tiempo, cada vez más complejas.

Aunque los distintos avances en la hibridación de la GP con otras técnicas, por ejemplo: las redes neuronales [51]; el uso de modelos ARIMA para la extracción de la componente lineal de la serie de tiempo estudiada, y luego, el uso de la GP para extraer, de los residuales del modelo ARIMA, la componente no lineal remanente [83]; el uso de reglas lingüísticas en la exploración del campo de búsqueda [8]; la inclusión de nuevos operadores [71]; y la redefinición de los individuos (incorporando el concepto de bloque funcional como constituyente básico) y operadores genéticos (agregando un parámetro aleatorio de agregación) para la focalización de los individuos en una zona específica de búsqueda durante la fase de exploración [12]. Han contribuido en la solución de los problemas de la aplicación de GP a la predicción de series temporales; aún persisten problemas de focalización adecuada de los individuos en regiones de interés de acuerdo a su estructura, la consistencia matemática de los individuos, la falta de parsimonia de los individuos, el aumento desmedido de su tamaño, la selección de aquellos individuos que realmente disminuyen el error de predicción por una estructura prometedora, y la inclusión de información exógena al modelo resultante; los cuales, representan interrogantes susceptibles a investigación, como son:

- *¿Es posible ampliar el conjunto de terminales a ser utilizados en el algoritmo de programación genética introduciendo información relevante que realmente influya positivamente en el error de aproximación del modelo resultante?*
  La selección de los terminales representa la base de la generación de los individuos, y por ende, determina en gran medida la exploración en el campo de búsqueda, por lo que la incorporación de nuevos terminales que realmente mejoren el valor de aptitud de los individuos (disminuyan el error de predicción) de la población es una necesidad constante en la implementación del algoritmo de GP.

- *¿Cómo incluir conocimiento experto para focalizar la búsqueda sobre los modelos estructuralmente más prometedores?*
  La falta de incorporación de conocimiento a priori en los terminales y operadores utilizados (funciones, rezagos, y parámetros de optimización), durante el proceso de búsqueda del algoritmo de GP (focalización del espacio de búsqueda), genera una búsqueda principalmente aleatoria, que no permite un mayor nivel de aproximación a la serie de tiempo modelada a un costo computacionalmente aceptable.

- *¿Es posible reducir las operaciones redundantes de los individuos sin afectar el proceso de exploración del algoritmo de programación genética?*
  Debido a que las modificaciones en el algoritmo de GP se han basado solo en modelos de agregación, sin incorporación de operaciones de simplificación de los mismos, la parsimonía de los individuos va en decremento durante la fase de exploración (aumento del bloat - aumento desmedido del tamaño de los individuos sin aporte a la reducción del error de predicción), lo que impacta negativamente en la calidad de soluciones encontradas [81, 82].

- *¿Es posible introducir conocimiento experto durante la generación de los individuos del algoritmo de programación genética que permita su consistencia lógico-matemática?*
  Debido a que la generación de los individuos del algoritmo original de GP, y sus modificaciones, se realiza aleatoriamente por medio de un proceso combinatorio de funciones, sin validación de la consistencia de los mismos, no es posible garantizar que dichos individuos (ecuaciones) sean

consistentes con los modelos de predicción ampliamente utilizados en la literatura de predicción de series de tiempo; ni tampoco es posible una equivalencia directa con las propiedades de la serie de tiempo analizada, ni una interpretación adecuada con respecto a las componentes principales de ciclo, tendencia, estacionalidad y error de una serie temporal.

1.5 Hipótesis

A partir del análisis realizado, se propone la siguiente hipótesis para esta tesis:

> *Es posible introducir conocimiento experto en la restricción y focalización del espacio de búsqueda del algoritmo de programación genética, garantizando el uso de todos los rezagos definidos en cada individuo de la población, y al menos, una de las funciones ampliamente utilizadas en la literatura de predicción de series de tiempo. Lo anterior, permitirá reducir el espacio de búsqueda, disminuir el costo computacional, y generar soluciones con menor error de predicción y mayor consistencia que las actualmente generadas por los algoritmos tradicionales de programación genética.*

1.6 Objetivos de la Tesis

1.6.1 Objetivo general

De acuerdo con las preguntas emergentes actuales de GP, a la hipótesis planteada, y las limitaciones presentadas en la discusión, se tiene como objetivo principal de esta investigación:

> *Proponer un algoritmo de generación de funciones de predicción de series de tiempo basado en el algoritmo de programación genética, que permita la inclusión de conocimiento experto durante el proceso de búsqueda.*

El logro de este objetivo general, permitirá la reducción del espacio de búsqueda, la disminución del costo computacional, y la generación de soluciones con menor error de predicción y mayor consistencia que las actualmente generadas por los algoritmos tradicionales de programación genética.

1.6.2 Objetivos específicos

Para lograr el objetivo general propuesto en la Subsección 1.6.1 se proponen los siguientes objetivos secundarios:

- Proponer un nuevo mecanismo de generación de individuos de programación genética por medio de agrupaciones de bloques funcionales, con el fin de garantizar individuos lógicos de manera matemática, y su posterior interpretación a la luz de las variables propias de modelación.
- Modificar el proceso de generación de los individuos de programación genética para garantizar la inclusión de cada uno de los rezagos de interés, con el fin de disminuir la influencia de los terminales en la selección de la estructura funcional de los individuos.
- Modificar los operadores genéticos para permitir la focalización del espacio de búsqueda del algoritmo de programación genética en zonas prometedoras de búsqueda.
- Implementar el algoritmo propuesto de programación genética, y validar sus bondades por medio de simulación contra series benchmark de la literatura de predicción de series de tiempo, comparando el error de predicción, la inclusión de los rezagos definidos, y la identificación de las componentes de los modelos de predicción de series de tiempo.

1.7 Aportes y contribuciones

Al lograr los objetivos propuestos en este trabajo se pretende:

- Disminuir el tamaño de los individuos de GP por medio del uso de bloques funcionales y la aplicación de criterios de simplificación, lo cual permite la generación de individuos más complejos, que reglejen la no linealidad de la serie temporal a menor costo computacional.
- Incluir las componentes funcionales de los principales modelos de predicción de series de tiempo de la literatura actual en la generación de los individuos de GP, permitiendo la hibridación automática de modelos de predicción de series temporales, a partir de componentes validados y comúnmente empleados en la literatura.
- Garantizar el uso de todos los rezagos definidos en cada individuo de GP durante el proceso de búsqueda del algoritmo de GP, lo cual permite que el modelo resultante sea seleccionado a partir de su aporte estructural a la predicción de la serie de tiempo y no a la inclusión de un rezago particular.
- Eliminar la dependencia entre la selección aleatoria de las constantes del conjunto de terminales, y el comportamiento del individuo durante el proceso de búsqueda del algoritmo de GP, lo cual implica la reducción del espacio de búsqueda y una selección de constantes más acordes a los datos de la serie de tiempo, no solo por simple aleatoriedad.
- Focalizar el espacio de búsqueda del algoritmo de GP, al inicio de su ejecución (a priori), por medio del aumento de la probabilidad de selección de bloques funcionales a partir de las características de la serie de tiempo; y durante el proceso de búsqueda, por medio de la modificación dinámica de las probabilidades de aplicación de los operadores genéticos, modificación del operador de selección, y la modificación de los operadores genéticos de cruce y mutación.

Lo anterior, permitirá mejorar la calidad de la solución encontrada en términos de su valor de aptitud, aumentando la parsimonía de los individuos —disminuyendo por ende el bloat— y una interpretación más cercana a la realidad del fenómeno.

1.8 Organización del documento

Con el fin de alcanzar los objetivos propuestos, abordando las preguntas emergentes, y cumpliendo con el alcance de la tesis, lo restante de este documento se organiza de la siguiente manera: en el Capítulo 2 se realiza un análisis de la metodología propuesta, incluyendo, la definición de los cambios en los bloques funcionales, la inclusión de estos en el algoritmo original de GP, el desarrollo de las mejoras propuestas al algoritmo de GP, y la verificación de las mejoras propuestas, evaluándolas contra series de tiempo con función de generación es conocida. En el Capítulo 3 se aplica la metodología planteada al análisis de cinco series benchmark de predicción de series de tiempo, comparando los resultados obtenidos contra los reportados en la literatura de predicción de series de tiempo. Finalmente en el Capítulo 4 se encuentran las conclusiones del trabajo realizado y recomendaciones de trabajo futuro. Cabe anotar que también se cuenta con un capítulo de Anexos, en el cual se incluye el algoritmo de programación genética original, la descripción detallada de los bloques funcionales y la implementación de las propuestas realizadas en este trabajo en el lenguaje R.

2. Metodología propuesta

Aunque en el trabajo realizado en la tesis de maestria de Martínez [12], se avanzó en la reducción del tamaño de los individuos por medio de la agrupación de subárboles a partir del uso de los BF incluidos en el conjunto de terminales; al analizar el proceso de búsqueda del algoritmo de GP se puede concluir que aún continua presentándose un aumento del tamaño de los individuos sin aporte a la aptitud —tambien llamado bloat—, inconsistencias matemáticas al realizarse combinaciones lineales de bloques funcionales asociados a componentes (Ciclo, Tendencia, Estacionalidad, Error) no compatibles, y la falta de focalización en zonas de interés durante la aplicación de los operadores genéticos actuales.

Para solucionar lo anterior, en esta tesis, se propusieron los primeros tres objetivos específicos, que corresponden a:

- *Proponer un nuevo mecanismo de generación de individuos de programación genética por medio de agrupaciones de bloques funcionales, con el fin de garantizar individuos lógicos de manera matemática, y su posterior interpretación a la luz de las variables propias de modelación.*
- *Modificar el proceso de generación de los individuos de programación genética para garantizar la inclusión de cada uno de los rezagos de interés, con el fin de disminuir la influencia de los terminales en la selección de la estructura funcional de los individuos.*
- *Modificar los operadores genéticos que permitan la focalización del espacio de búsqueda del algoritmo de programación genética en zonas prometedoras de búsqueda*.

Para lograr los anteriores objetivos específicos, en este capítulo son presentados los cambios propuestos al algoritmo de GP (algoritmo original y modificaciones planteadas en la tesis de maestría de Martínez [12]), los cuales corresponden a: el cambio del conjunto de terminales (agrupando los bloques funcionales de acuerdo a las componentes de las series de tiempo); y las modificaciones de la función de aptitud, los pasos del algoritmo y los operadores de reproducción (clonación), mutación, cruce, selección e intensificación.

Los cambios propuestos en este capitulo permiten reducir la redundancia en los operadores —reducciendo el bloat—; mejorar la convergencia del modelo resultante con respecto a la serie de tiempo analizada (menor error de predicción); focalizar el espacio de búsqueda de GP en zonas de interés; la inclusión de conocimiento externo; y la inclusión de funciones ampliamente utilizadas en la literatura de predicción de series de tiempo, que permitan una interpretación más fácil y directa con las propiedades y componentes de la serie temporal, con una mayor consistencia matemática entre las componentes del modelo resultante de predicción.

Adicionalmente, se propone una medida del aporte de la estructura funcional de los individuos al valor de la aptitud de los mismos, independiente de las constantes seleccionadas en el conjunto de terminales; la modificación de la función de aptitud, por medio de la inclusión de criterios de tamaño de los individuos, y la inclusión de pasos adicionales de simplificación de la estructura de los individuos al proceso de búsqueda de GP, que permitan la disminución del crecimiento del tamaño desmedido de los individuos (bloat) durante el proceso de búsqueda del algoritmo de GP.

De acuerdo a lo anterior, en la Sección 2.1 se analizan los cambios referentes al individuo de GP; en la Sección 2.2 los cambios asociados al algoritmo; en la Sección 2.3 se validan los cambios planteados por medio

del análisis de series de tiempo con función de generación conocida. Por último, en la Sección 2.4 se plantean las respectivas conclusiones del capítulo.

2.1 Modificaciones a los individuos de programación genética

El modelamiento y predicción de series de tiempo busca encontrar aquella expresión matemática $\hat{F}(\mathbf{x}, \boldsymbol{\theta})$ que aproxime el comportamiento de la serie temporal, tal que: $\hat{F}(\mathbf{x}, \boldsymbol{\theta}) = F(\mathbf{x}, \boldsymbol{\theta}) + \delta$, donde $\delta \epsilon \mathbb{R}$ corresponde al error de la predicción, tal que el valor esperado de δ es igual a 0 ($\mathrm{E}[\delta] = 0$); $\mathbf{x} = \{x_{t-1}, \dots, x_{t-p}\}$ es el vector de rezagos a ser considerados; $\boldsymbol{\theta} = \{\theta_1, \dots, \theta_m\}$ es el vector de parámetros y coeficientes; y $F(\mathbf{x}, \boldsymbol{\theta})$ corresponde a la serie de tiempo original.

En la tesis de maestría [12] se analizó la serie temporal bajo una definición de una única función $F(\mathbf{x}, \boldsymbol{\theta})$, lo que conlleva a que se realicen combinaciones lineales entre nodos (bloques funcionales para la propuesta realizada) durante el proceso de búsqueda del algoritmo de GP (debido a la aplicación aleatoria de los operadores genéticos de cruce y mutación), que no son consistentes matemáticamente (mezcla de componentes de la serie de tiempo que carece de lógica matemática), lo que impide su asociación y coherencia con la realidad del fenómeno analizado. Para solventar este problema, en este trabajo se analizan las series de tiempo a partir de sus componentes de: estacionalidad, ciclo, tendencia y error, permitiendo describir la serie de tiempo de una manera funcional, por medio de la expresión matemática:

$$F(\mathbf{x}, \boldsymbol{\theta}) : \{[\mathbf{x} \in \mathbb{R}^p, \boldsymbol{\theta} \in \mathbb{R}^m] \rightarrow \mathbb{R} \mid F(\mathbf{x}, \boldsymbol{\theta}) = \lambda_1 S(\mathbf{x}, \boldsymbol{\theta}) + \lambda_2 C(\mathbf{x}, \boldsymbol{\theta}) + \lambda_3 \Gamma(\mathbf{x}, \boldsymbol{\theta}) + \lambda_4 \Upsilon(\mathbf{x}, \boldsymbol{\theta})\} \tag{2.1}$$

Donde $S(.)$ es la componente de estacionalidad, $C(.)$ es la componente de ciclo, $\Gamma(.)$ es la componente de tendencia, $\Upsilon(.)$ es la componente de error, y $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$ son coeficientes dicotómicos $\lambda_{1 \leq i \leq 4} \in [0,1]$ que indican si aplica o no la componente de acuerdo con restricciones matemáticas.

Tradicionalmente el vector de parámetros $\boldsymbol{\theta} \in \mathbb{R}^m$ es asumido implícitamente, por lo que la ecuación 2.1 es equivalente a la expresión:

$$F(\mathbf{x}) : \{\mathbf{x} \in \mathbb{R}^p \rightarrow \mathbb{R} \mid F(\mathbf{x}) = \lambda_1 S(\mathbf{x}) + \lambda_2 C(\mathbf{x}) + \lambda_3 \Gamma(\mathbf{x}) + \lambda_4 \Upsilon(\mathbf{x})\} \tag{2.2}$$

En la Figura 2-1 se muestran las componentes de ciclo, tendencia y error, para la serie *Births* (número mensual de aves en New York entre ENE1946 y DIC1959). Se puede apreciar que cada componente equivale a una función susceptible de ser modelada independientemente [76].

De acuerdo a la ecuación 2.1, el conjunto de funciones $\{S(.), C(.), \Gamma(.), \Upsilon(.)\}$ constituyen la forma de la función $F(.)$; mientras que, el vector de parámetros $\boldsymbol{\theta}$ corresponde a las respectivas perturbaciones y ajuste del modelo; es decir, para una función $F(.)$, bajo los mismos criterios de optimización de sus parámetros (ver la Subsección 2.2.6), la mejora en el valor de aptitud (fitness) estará asociada a la estructura de $F(.)$.

Para hallar la ecuación $\hat{F}(.)$, el algoritmo de GP utiliza una serie de pasos que se fundamentan en la aplicación de operadores genéticos a una población de individuos inicial generada aleatoriamente [3, 14]; dichos individuos son ecuaciones matemáticas representadas por arboles sintácticos, en los cuales los nodos corresponden a operadores (se suele utilizar operaciones aritméticas básicas como $Z = \{+, -, /, *\}$) y las hojas a los terminales (rezagos y constantes a ser utilizados) [3, 14]; lo que implica que los terminales son constituidos por cada uno de los rezagos y constantes a ser utilizadas (estáticas y dinámicas) [14], y la selección de los terminales a ser incluidos en los individuos corresponde a un proceso principalmente aleatorio (los operadores genéticos de mutación y reproducción se basan en la selección aleatoria de nodos e individuos para su aplicación [14]), por lo que no es posible garantizar el uso de todos los rezagos definidos en los individuos de GP durante el proceso de búsqueda, ni una jerarquía de generación por componente de la serie [8].

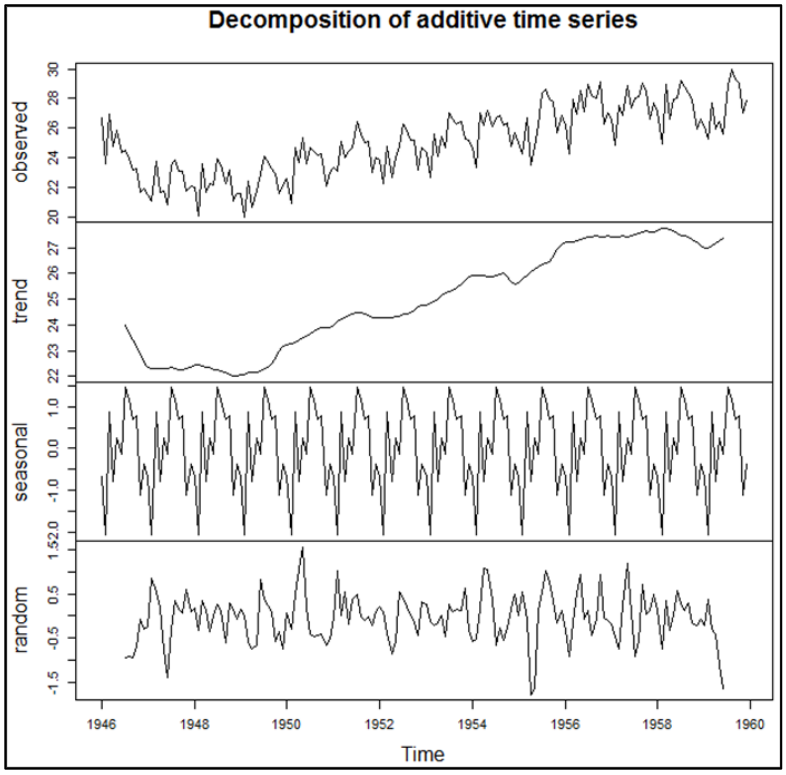


Figura 2-1: Representación de las componentes de la serie Births.

Para solventar el problema anterior, mejorando el valor de aptitud (disminución del error de predicción) y aumentando la consistencia matemática de la solución encontrada por GP, es necesario reescribir los individuos en términos de las componentes de estacionalidad, ciclo, tendencia y error de la serie temporal, por lo que se modifican los terminales de los individuos de GP para incluir dichos componentes, categorizando los bloques funcionales de acuerdo al tipo de componente asociado (ciclo, tendencia, estacionalidad y error), equivalente a la ecuación 2.2, generando así, un conjunto de reglas para el cálculo del vector $\boldsymbol{\lambda}$ que garanticen su consistencia matemática durante la generación de la población inicial y la aplicación de los operadores genéticos. Adicionalmente, se propone la inclusión de un vector de parámetros generales y variables exógenas ($\boldsymbol{\theta}$) de relevancia para la limitación del espacio de búsqueda del algoritmo de GP, modificando los bloques funcionales para la incorporación del vector $\boldsymbol{\theta}$.

Por lo que, en esta subsección serán desarrolladas las propuestas mencionadas en cada uno de sus numerales.

2.1.1 Modificación de los bloques funcionales para la inclusión de modelos de predicción de series de tiempo

Los bloques funcionales son los constituyentes básicos de las ecuaciones; corresponden a las funciones más básicas que se pueden manejar como terminales en un individuo de GP. De esta forma, los BF pueden ser evaluados numéricamente sin depender de otras funciones externas. Todas las funciones definidas para los nodos interiores del árbol sintáctico pueden operar directamente sobre los BF sin modificaciones, debido a que los BF ocupan únicamente los nodos terminales.

En su forma más simple, los BF pueden ser definidos como una función $B(\cdot)$ que corresponde a la combinación lineal de las funciones $G_i(\cdot)$:

$$B(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{i=1} w_i G_i(\mathbf{x}, \boldsymbol{\theta}) \tag{2.3}$$

Donde $\mathbf{x} = \{x_{t-1}, \dots, x_{t-p}\}$ es el vector de entradas (variables/rezagos) del modelo, $\boldsymbol{\theta}$ es el vector de parámetros generales del BF, $G_i(\cdot)$ son las funciones generadoras del BF, y $\mathbf{w}$ es el vector de coeficientes de las funciones $G_i(\cdot)$. Adicionalmente, es posible definir BF más complejos a partir de la combinación de BF más simples, reemplazando la función $G_i(\cdot)$ por un bloque funcional.

Tradicionalmente el vector de parámetros $\mathbf{w} \in \mathbb{R}^m$ es asumido implícitamente, por lo que la ecuación 2.3, es equivalente a la expresión:

$$B(\mathbf{x}) = w_0 + \sum_{i=1} w_i G_i(\mathbf{x}) \tag{2.4}$$

Adicionalmente, por claridad de notación, se asumirá que $B_i = B_i(.)$ y $G_i = G_i(.)$, por lo que la ecuación 2.3 se puede reducir a la expresión:

$$B = w_0 + \sum_{i=1} w_i G_i \tag{2.5}$$

A continuación se ejemplifica el uso de los BF. Sea, por ejemplo, el modelo autoregresivo de media móvil $ARMA(p, q)$, con p número de rezagos y q número de diferencias, definido como: $x_t = \sum_{i=1}^{p} \varphi_i x_{t-i} + \sum_{i=1}^{q} \alpha_i \varrho_{t-i} + \varepsilon_t$, donde $\boldsymbol{\varphi}$ es el vector de coeficientes del modelo autoregresivo, $\boldsymbol{\alpha}$ es el vector de coeficientes del modelo de media móvil, $\mathbf{x}$ es el vector de rezagos, $\boldsymbol{\varrho}$ es el vector de diferencias del modelo de media móvil, y $\boldsymbol{\varepsilon}$ es el vector de errores. Si asumimos el bloque funcional B_1 como la parte autoregresiva (AR) del modelo (con $G_i(\mathbf{x}, \mathbf{w}) = \mathbf{x}_{t-i}$), el bloque funcional B_2 como la componente de medias móviles (MA) del modelo (con $G_i(\mathbf{x}, \mathbf{w}) = \boldsymbol{\varrho}_{t-i}$), y el bloque funcional B_3 equivalente a la componente de error del modelo ($G_i(\mathbf{x}, \boldsymbol{\theta}) = \boldsymbol{\varepsilon}_t$), el modelo $ARMA(p, q)$ es equivalente a:

$$x_t = \sum_{i=1}^{p} \varphi_i x_{t-i} + \sum_{i=1}^{q} \alpha_i \varrho_{t-i} + \varepsilon_t = B_1 + B_2 + B_3 \tag{2.6}$$

Entonces el modelo anterior x_t puede ser representado como la suma de tres bloques funcionales y equivale al árbol sintáctico presentado en la Figura 2-2.

Entre las principales características de los BF se encuentran las siguientes:

- Permiten expresar ecuaciones complejas de forma simple, tal como ya se ilustró; ello facilita la aplicación de los operadores genéticos, evita la redundancia de operadores, (en la versión clásica de GP serían necesarios $2 * (p + q) + 1$ nodos, mientras en la versión con BF son sólo necesarios cinco nodos) y permite una exploración más amplia del campo de búsqueda.
- Los individuos resultantes pueden interpretarse como modelos híbridos que combinan de forma novedosa los modelos básicos definidos por los BF.
- Los individuos pueden interpretarse en términos de los BF que los conforman, y por consiguiente, en términos de los modelos bien establecidos en la literatura, logrando mayor claridad.

En el algoritmo original de GP, se toma un único conjunto de terminales $T = \{x_{t-1}, \dots, x_{t-p}\} \cup \{w_1, \dots, w_l\} = \{\mathbf{x} \cup \mathbf{w}\}$, y operadores $\mathrm{Z} = \{\zeta_1, \dots, \zeta_k\}$, para la generación aleatoria del individuo I, el cual se muestra en el Algoritmo 2.1.

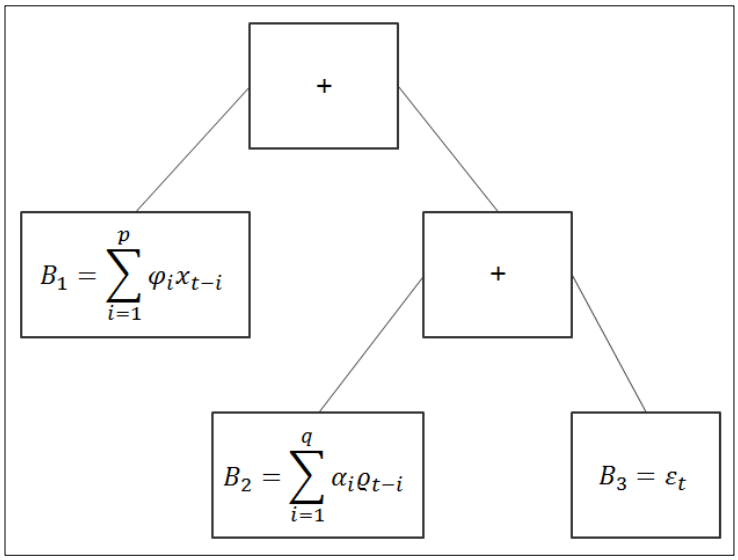


Figura 2-2: Representación de un modelo ARMA por medio de bloques funcionales.

En el Algoritmo 2.1, se utiliza la función $Node(I, j)$ la cual retorna el nodo j del individuo I; la función $random(a, b)$ que retorna un número aleatorio real entre a y b, la función $ceil(a)$ que retorna un número entero correspondiente al techo de a de acuerdo con la expresión: $ceil(a) = min\{n \in \mathbb{Z} \mid n \geq a\}$; y la función $numArg(op)$ que retorna el número de argumentos que recibe el operador op; por ejemplo: $numArg(op = \{+\}) = 2$.

Algoritmo 2.1. Generación aleatoria de individuos – Algoritmo original de GP.

```
01  H_max
02  ϰ_max
03  T ← {x_{t−1}, … , x_{t−p}} ∪ {w_1, … , w_{l*}}
04  Z ← {ζ_1, … , ζ_k}
05  I ← { }
06  H ← 0
07  ϰ ← 0
08  loop while H ≤ H_max and ϰ ≤ ϰ_max and {∃j ∈ ℕ+ | Node(I, j) ⊈ T}
09          a ← random(0,1)
10          if a ≤ 0.5 then
11                  Node(I, j) ← T_{ceil(random(1,p+l))}
12          else
13                  Node(I, j) ← Z_{ceil(random(1,k))}
15                  for k =1 to numArg(Node(I, j)) do
16                          Node(I, j + k) ← { }
17                  end for
18                  H ← H + 1
19          end if
20          ϰ ← ϰ + 1
21  end loop
22  end algorithm
```

En la propuesta desarrollada en la tesis de maestría por Martínez [12], se plantea un único vector terminales $T = \{B_1, \dots, B_l, c_1, \dots\}$, constituido por todos los bloques funcionales y las constantes reales a ser incluidas, sin restricciones en la combinación de los mismos, ni aumento de la probabilidad de uso de acuerdo

con las características de la serie de tiempo, equivalente al Algoritmo 2.1 modificando la línea 3 por: $T \leftarrow \{B_1, \dots, B_l\}$, manteniendo todos los demás pasos iguales. Lo anterior implica que no existe garantía en el uso de modelos de predicción de series de tiempo ampliamente utilizados en la literatura, ni una consistencia matemática entre las distintas componentes de la serie de tiempo (ciclo, tendencia, estacionalidad y error), las cuales son falencias de las implementaciones de GP a la predicción de series de tiempo que deterioran la calidad de la solución encontrada.

Considerando que un individuo I de GP es un modelo de predicción de series de tiempo, y que las series de tiempo corresponden a una función $F(.)$, es posible afirmar que un individuo es por ende una aproximación $\hat{F}(.)$ de $F(.)$; y, de acuerdo con ecuación 2.2, se propone reescribir a los individuos por medio de la expresión:

$$I(\mathbf{x}, \boldsymbol{\theta}) = \hat{F}(\mathbf{x}, \boldsymbol{\theta}) = F(\mathbf{x}, \boldsymbol{\theta}) + \delta \tag{2.7}$$

donde, $\delta \in \mathbb{R}$ pequeño tal que el $E[\delta] = 0$, $\hat{F}(\mathbf{x}, \boldsymbol{\theta}) \approx F(\mathbf{x}, \boldsymbol{\theta})$ con el $E\left[\left|\hat{F}(\mathbf{x}, \boldsymbol{\theta}) - F(\mathbf{x}, \boldsymbol{\theta})\right|\right] = 0$; por lo que es posible reescribir a un individuo I como una combinación lineal de las componentes de las series de tiempo, de acuerdo con la expresión:

$$I(\mathbf{x}, \boldsymbol{\theta}) = \lambda_1 S(\mathbf{x}, \boldsymbol{\theta}) + \lambda_2 C(\mathbf{x}, \boldsymbol{\theta}) + \lambda_3 \Gamma(\mathbf{x}, \boldsymbol{\theta}) + \lambda_4 \Upsilon(\mathbf{x}, \boldsymbol{\theta}) \tag{2.8}$$

Tradicionalmente, el vector de parámetros $\boldsymbol{\theta} \in \mathbb{R}^m$ y el vector de rezagos $\mathbf{x} \in \mathbb{R}^p$, son asumidos implícitamente en la notación del individuo, por lo que la ecuación 2.8, es equivalente a la expresión:

$$I = \lambda_1 S + \lambda_2 C + \lambda_3 \Gamma + \lambda_4 \Upsilon \tag{2.9}$$

Considerando que la forma de la función $F(.)$ está determinada por la relación entre sus variables de entrada, por medio del conjunto de componentes $\{S(.), C(.), \Gamma(.), \Upsilon(.)\}$, también llamada forma funcional; que el vector de parámetros $\boldsymbol{\theta}$ corresponde a las perturbaciones y ajustes de $F(.)$ (marco de referencia); que de acuerdo con la ecuación 2.7, los individuos I de programación genética corresponden a aproximaciones de $F(.)$; y que los modelos de predicción de series temporales corresponden a versiones de $\hat{F}(.)$; es posible incluir los modelos de la literatura de predicción de series de tiempo a los individuos I por medio del uso de bloques funcionales.

De acuerdo a la ecuación 2.4, los bloques funcionales $B(.)$ son expresados por medio de una combinación lineal de funciones más simples $G(.)$; al analizar la literatura actual de predicción de series temporales, es posible expresar cada uno de los modelos actuales en forma de sus componentes $G(.)$, equivalentes por ende a los bloques funcionales. En la Tabla 2-2 se encuentran consignados los distintos modelos analizados, con su respectiva expresión con bloques funcionales. Debido a que muchos de los bloques funcionales son reiterativos en distintos modelos, en la Tabla 2-1 se encuentran los bloques definidos con los cuales fue construida la Tabla 2-2, los cuales serán utilizados a lo largo del documento.

Adicionalmente, debido a que los BF agrupan funciones predefinidas, las cuales pueden corresponder a las componentes de los modelos de predicción de series de tiempo presentes en la literatura, y que durante el proceso de búsqueda del algoritmo de GP se realizan combinaciones lineales y no lineales de los nodos (pertenecientes al conjunto de terminales T) de los individuos de la población, se puede afirmar que el algoritmo de GP propuesto en esta tesis (incorporando los bloques funcionales como terminales), contituye un algoritmo de hibridación automática de funciones de predicción de series de tiempo, cuya calidad esta sujeta al proceso de exploración del espacio de búsqueda y la función de aptitud empleada.

En la Tabla 2-2 se muestra que es posible incorporar, a partir de los bloques funcionales definidos en la Tabla 2-1, los principales modelos de la literatura de predicción de series de tiempo, siendo posible su posterior análisis en en agrupaciones de acuerdo con las componentes $\{S, C, \Gamma, \Upsilon\}$, permitiendo una mayor consistencia

de los individuos generados durante el proceso de búsqueda, logrando contribuir al logro del primer objetivo específico.

Tabla 2-1: Bloques funcionales identificados en la literatura.

BF	Ecuación	Parámetros (θ)
B_1	$\sum_{i=1}^{p} w_i x_{t-i}$	$p \in \mathbb{Z} \mid p \geq 1$ es el número de rezagos. w_i son constantes reales a encontrar.
B_2	$\sum_{i=1}^{q} w_i \varepsilon_{t-i}$	$q \in \mathbb{Z} \mid q \geq 1$ es el número de diferencias. ε_{t-i} es la diferencia entre los rezagos x_{t-i} y x_{t-i-1}. w_i son constantes reales a encontrar.
B_3	ε_t	ε_t iid de preferencia $N(0, \sigma^2)$ para el tiempo t.
B_4	$\sum_{i=1}^{p} \{w_i x_{t-i}(r_i + s_i x_{t-h})\} = \sum_{i=1}^{p} \{r_i x_{t-i} + s_i x_{t-i} x_{t-h}\}$	$p \in \mathbb{Z} \mid p \geq 1$ es el número de rezagos. w_i, r_i, s_i son constantes reales a encontrar.
B_5	$\sum_{i=1}^{q} \{w_t \varepsilon_{t-i}(u_i + v_i \varepsilon_{t-k})\}$	$q \in \mathbb{Z} \mid q \geq 1$ es el número de diferencias. ε_{t-i} es la diferencia entre los rezagos x_{t-i} y x_{t-i-1}. w_i, u_i, v_i son constantes reales a encontrar.
B_6	$\sigma^2 \varepsilon_t$	ε_t iid de preferencia $N(0, \sigma^2)$. σ^2 corresponde a la varianza.
B_7	c	c es una constante real a ser encontrada.
B_8	$F\left(\frac{x_{t-d} - \Delta}{s}\right)$	$F(.)$ es la función de transición; puede ser logística, exponencial, distribución acumulativa, entre otras.
B_9	$\sum_{i=1}^{p} \varphi_i K(x_{t-i})$	$K(.)$ es una función tipo kernel.
B_{10}	$\sum_{h=1}^{H} \beta_h G\left((2\sigma_t)^{-1} \alpha_{0h} + \sum_{i=1}^{I} \alpha_{ih} x_{t-i}\right)$	$G(.)$ es la función de activación de las neuronas de la capa oculta en una red neuronal auto regresiva (ARNN). σ corresponde a la desviación estándar de los errores. ε_t es una variable aleatoria que sigue una distribución normal estándar. H es el número de neuronas en la capa oculta. I es el número de regresores. σ_t es la desviación estándar de x_t; su uso evita tener que transformar x_t para restringir sus valores al rango de la función G(.).
B_{11}	$\left(1 - \sum_{i=1}^{p} \varphi_i L^i (r_i + s_i L^h)\right)$	$p \in \mathbb{Z} \mid p \geq 1$ es el número de rezagos. $L^i = x_{t-i}$. φ_i, r_i, s_i son constantes reales (pueden estar dadas o susceptibles a ser encontradas).
B_{12}	$\left(1 + \sum_{i=1}^{q} w_i L^i (u_i + v_i L^k)\right)$	$q \in \mathbb{Z} \mid q \geq 1$ es el número de diferencias. $L^i = x_{t-i}$. w_i, u_i, v_i son constantes reales (pueden estar dadas o susceptibles a ser encontradas).
B_{13}	$(1 - L)^d$	$L = x_t$, d es el factor de integración.
B_{14}	$\Phi_s(B^s)$	$\Phi_s(B^s)$ Polinomio con periodo s. $B^s = L^s = x_{t-s}$.
B_{15}	$\varphi(B)$	$\varphi(B)$ Polinomio con periodo s, con $B = L = X_t$.
B_{16}	$(1 - B^s)^d$	$B^s = L^s = x_{t-s}$.
B_{17}	$H_s(B^s)$	$H_s(B^s)$ Polinomio con periodo s, con $B^s = L^s = x_{t-s}$.
B_{18}	$h(B)$	$h(B)$ Polinomio con periodo s, con $B = L = x_t$.
B_{19}	$e^{-p_i X_{t-d}^2}$	Es parte de una función típica de kernel.
B_{20}	$2\pi^{-\frac{p}{2}} e^{-\frac{\|x^2\|}{2}}$	Utilizada como función de kernel que corresponde a su función normal de densidad.
B_{21}	$C_{k,p} k(\|x\|)$	Utilizada como función de kernel conocida como "kernel esférico simétrico" donde: $C_{k,p} = \{\int K(\|x\|)dx\}^{-1}$ y $\|x\| = \left(x_1^2 + \cdots + x_p^2\right)^{1/2}$.
B_{22}	$\prod_{i=1}^{p} k(x_i)$	Corresponde al producto kernel donde k() corresponde a la función de kernel seleccionada.

Tabla 2-1: Bloques funcionales identificados en la literatura (Continuación).

BF	Ecuación	Parámetros (θ)
B_{23}	X_t	x_t corresponde al rezago en el tiempo t.
B_{24}	$g_i(\beta^T x_{t-d})$	Familia de funciones auto regresivas en función del rezago $t-d$ de x y una dirección del modelo dependiente β^T.
B_{25}	$\sum_{i=1}^{p} \varphi_i x_{t-i} + c$	$p \in \mathbb{Z} \mid p \geq 1$ es el número de rezagos φ_i, c son constantes reales a encontrar.
B_{26}	$\frac{\left(\sum_{i=1}^{p} \varphi_i x_{t-i} + \sum_{i=1}^{q} w_i \varepsilon_{t-i} + c\right)}{(1 + x_{t-1})^d}$	Representación de un modelo $Arima(p,d,q)$.
B_{27}	$\sum_{i=1}^{p} \varphi_i \ sin(x_{t-i})$	$p \in \mathbb{Z} \mid p \geq 1$ es el número de rezagos φ_i, c son constantes reales a encontrar.
B_{28}	$\sum_{i=1}^{p} \varphi_i \ cos(x_{t-i})$	$p \in \mathbb{Z} \mid p \geq 1$ es el número de rezagos φ_i, c son constantes reales a encontrar.
B_{29}	$\sum_{i=1}^{p} \varphi_i \ exp(x_{t-i})$	$p \in \mathbb{Z} \mid p \geq 1$ es el número de rezagos φ_i, c son constantes reales a encontrar.
B_{30}	$\sum_{i=1}^{p} \varphi_i / \left(1 - exp(-x_{t-i})\right)$	$p \in \mathbb{Z} \mid p \geq 1$ es el número de rezagos φ_i, c son constantes reales a encontrar.

Tabla 2-2: Representación de los modelos de predicción de series de tiempo seleccionados por medio de bloques funcionales.

Modelo	Ecuación	Ecuación con BF
AR [13]	$x_t = \sum_{i=1}^{p} \varphi_i x_{t-i} + \varepsilon_t$	$x_t = B_1 + B_3$
MA [13]	$x_t = \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t$	$x_t = B_2 + B_3$
ARMA [13]	$x_t = \sum_{i=1}^{p} \varphi_i x_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t$	$x_t = B_1 + B_2 + B_3$
ARIMA [13]	$\left(1 - \sum_{i=1}^{p} \varphi_i L^i\right)(1-L)^d X_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \varepsilon_t$	$X_{x_t} = B_{26}$
SARIMA [13]	$\Phi_s(B^s)\varphi(B)(1-B)^d(1-B^s)^d X_t = H_s(B^s)h(B)\varepsilon_t$	$B_{14} * B_{15} * B'_{16} * B_{16} * B_{23} = B_{17} * B_{18} * B_3 B'_{16} = B_{16}$ con $s = 1$.
NARX [84]	$x_t = F\left(x_{t-1}, \dots, x_{t-p}, u_{t-1}, \dots, u_{t-q}\right) + \varepsilon_t$	Dependiendo de la forma de $F(.)$ se generarían sus bloques funcionales.
VAR [85]	$x_t = c + \sum_{i=1}^{p} A_i x_{t-i} + \varepsilon_t$	$x_t = B_7 + B_1 + B_3$
TAR [86, 87]	$x_t = \sum_{i=1}^{n+1} \gamma_i g_i(x_{t-i}) = Bloque_9$ $\gamma_1 = \begin{cases} 1 \ si \ x_{t-i} \leq Umbral_1 \\ 0 \ en \ otro \ caso \end{cases}$ $\gamma_i = \begin{cases} 1 \ si \ x_{t-i} \leq Umbral_i \ y \ \gamma_j = 0 \ para \ \forall_{j<i} \\ 0 \ en \ otro \ caso \end{cases}$	$x_t = B_9$
SETAR [86, 88-93]	$x_t = \sum_{i=1}^{p} \phi_i^{(j)} x_{t-p} + a_t^{(j)}$	$x_t = B_1 + B_7$
ARCH [94]	$x_t = \sigma_t \varepsilon_t$ $\sigma_t^2 = c + \sum_{i=1}^{p} b_i x_{t-i}^2$	$x_t = B'_6$ $\sigma_t^2 = B_7 + B''_1$ $B'_6 = B_6$ con $\sigma_t^2 = \sigma_t$ $B''_1 = B_1$ con $x_{t-i} = x_{t-i}^2$

Tabla 2-2: Representación de los modelos de predicción de series de tiempo seleccionados por medio de bloques funcionales (Continuación).

Modelo	Ecuación	Ecuación con BF
GARCH [95]	$x_t = \sigma_t$ $\sigma_t^2 = c + \sum_{i=1}^{p} b_i x_{t-i}^2 + \sum_{i=1}^{q} a_i \sigma_{t-i}^2$	$x_t = \mathrm{B}_6'$ $\sigma_t^2 = B_7 + B_1'' + B_2''$ $\mathrm{B}_6' = B_6$ con $\sigma_t^2 = \sigma_t$ $B_1'' = B_1$ con $x_{t-i} = x_{t-i}^2$ $B_2'' = B_2$ con $\sigma_{t-i} = \sigma_{t-i}^2$,
MLP [96]	$\int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty} k(x)dx = 1$ $k(x) = \begin{cases} \prod_{i=1}^{p} k(x_i) \\ C_{k,p} k(\lVert x \rVert) \\ 2\pi^{-\frac{p}{2}} e^{-\frac{\lVert x^2 \rVert}{2}} \end{cases}$	$\mathrm{k(x)} = \begin{cases} \mathrm{B}_{22} \\ \mathrm{B}_{21} \\ \mathrm{B}_{20} \end{cases}$
FAR [97]	$x_t = \sum_{i=1}^{p} a_i x_{t-d} x_{t-i} + x_{t-d}\sigma\varepsilon_t$	$X_t = 1 - B_{11}'' + B_8' * \mathrm{B}_6'$ $B_8' = \mathrm{B}_8$ con $\Delta = 0, s = 1$ $B_{11}'' = \mathrm{B}_{11}$ con $r_i = 0, s_i = 1, h = d$
EXPAR [98]	$x_t = \sum_{i=1}^{p} \left(\alpha_1 + \{\beta_i + \Upsilon_i x_{t-d}\} e^{-p_i X_{t-d}^2}\right) x_{t-i} + \varepsilon_t$ $= \sum_{i=1}^{p} \alpha_1 x_{t-i} + \sum_{i=1}^{p} \{\beta_i + \Upsilon_i x_{t-d}\} e^{-p_i X_{t-d}^2} x_{t-i} + \varepsilon_t$	$X_t = B_1'' + 1 - B_{11}''' + B_3$ $B_1'' = \mathrm{B}_1$ con $\varphi_i = \alpha_1$ $B_{11}''' = \mathrm{B}_{11}$ con $\varphi_i = B_{19}$, $r_i = \beta_i$, $s_i = \Upsilon_i$
AFAR [99]	$x_t = g_0(\beta^T x_{t-1}) + \sum_{i=1}^{p} g_i(\beta^T x_{t-1}) x_{t-i} + \varepsilon_t$	$x_t = \mathrm{B}_{24}' + \mathrm{B}_1''' + \mathrm{B}_3$ $\mathrm{B}_{24}' = \mathrm{B}_{24}$ con $i = 0$ y $d = 1$ $\mathrm{B}_1''' = \mathrm{B}_1$ con $\varphi_i = B_{24}$ con $d = 1$
BL [100-103]	$x_t = \mu + \sum_{i=1}^{q} \beta_i \varepsilon_{t-i} \varepsilon_t + \varepsilon_t$	$x_t = \mathrm{B}_7 + \mathrm{B}_5'' + B_3$ $B_5'' = \mathrm{B}_5$ con $u_i = 0$ y $v_i = 1$
STAR [104-106]	$x_t = c_0 + \sum_{i=1}^{p} \varphi_i x_{t-i} + F\left(\frac{x_{t-d} - \Delta}{s}\right)\left(c_1 + \sum_{i=1}^{p} \theta_i x_{t-i}\right) + \varepsilon_t$	$x_t = B_7 + B_1 + B_8 * (B_7 + B_1) + B_3$
FUZZY [107]		Equivalente a un STAR
ARNN [108, 109]	$x_t = \beta_0 + \sum_{i=1}^{p} \varphi_i x_{t-i} + \sum_{h=1}^{H} \beta_h G\left((2\sigma_t)^{-1}\alpha_{0h} + \sum_{i=1}^{I} \alpha_{ih} x_{t-i}\right) + \varepsilon_t$	$x_t = B_7 + B_1 + B_{10} + B_3$
GENERAL ARMA	$\left(1 - \sum_{i=1}^{p} \varphi_i L^i (r_i + s_i L^h)\right) x_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i (u_i + v_i L^k)\right) \sigma^2 \varepsilon_t + c$	$B_{10} * B_{23} = B_{11} * B_6 + B_7$
GENERAL ARIMA	$\left(1 - \sum_{i=1}^{p} \varphi_i L^i (r_i + s_i L^h)\right)(1 - L)^d x_t$ $= \left(1 + \sum_{i=1}^{q} \theta_i L^i (u_i + v_i L^k)\right) \sigma^2 \varepsilon_t + c$	$B_{10} * B_{13} * B_{23} = B_{11} * B_6 + B_7$
GENERAL ARNN	$x_t = \beta_0 + F\left(\frac{x_{t-d} - \Delta}{s}\right) \sum_{i=1}^{p} \varphi_i K(x_{t-i})$ $+ \sum_{h=1}^{H} \beta_h G\left((2\sigma_t)^{-1}\alpha_{0h} + \sum_{i=1}^{I} \alpha_{ih} x_{t-i}\right) + \varepsilon_t$	$x_t = B_7 + B_8 * B_9 + B_{10} + B_3$

2.1.2 Eliminación de los terminales tipo constante y adición de todos los rezagos definidos en la generación de los individuos

En el algoritmo original de GP, se toma un conjunto de operadores Z y terminales T para la generación aleatoria de los individuos. El vector de terminales T, lo constituye los distintos rezagos y constantes a ser utilizadas como hojas del árbol sintáctico que representa el individuo I, mientras que Z suele ser el conjunto de operadores aritméticos básicos ($\{+, -, \times, \div\}$). Por otra parte, en la propuesta realizada en a tesis de maestría por Martínez [12] son cambiados los rezagos por bloques funcionales.

Debido a que la generación de los individuos es realizada de manera meramente aleatoria (como se muestra en el Algoritmo 2.1 de la Subsección 2.1.1), no es posible garantizar el uso de todos los rezagos definidos durante la generación de los individuos; y teniendo en cuenta la alta dependencia entre la salida del modelo y la correcta selección de sus entradas (rezagos en este caso), la no inclusión de rezagos relevantes va en detrimento de la aptitud de los individuos durante el proceso de búsqueda del algoritmo de GP.

Por otra parte, la no identificación de estructuras funcionales de interés, debido a que dos estructuras funcionales similares con rezagos distintos generarán valores de aptitud diferentes, dado que cada individuo es una función de los terminales T, por lo que si existe una alta correlación entre un terminal específico y las salida, y éste no es incluido en el individuo —siendo estadísticamente independiente de los demás rezagos— el valor de la aptitud de dicho individuo será menor al de cualquier otro con dicho terminal.

Adicionalmente, las constantes (tanto estáticas como dinámicas), las cuales son introducidas como terminales para la optimización de la estructura funcional del individuo [14], genera que los individuos crezcan desmedidamente sin aportar a la predicción de la serie de tiempo de una manera efectiva —también llamado bloat— lo que genera problemas de convergencia hacia soluciones de interés y un consumo excesivo de recursos computacionales durante el proceso de búsqueda del algoritmo de GP [81, 82]. Por ejemplo, en la Figura 2-3 se muestra el individuo $I = 1 * 1 * 1 * 1 * x$ y su respectiva simplificación unificando constantes, cuyo resultado es $I = x$.

Por lo anterior, y para solucionar los problemas de falta de inclusión de los rezagos de interés en la generación de los individuos y su crecimiento desmedido a causa de la agregación reiterativa de constantes, en este trabajo se propone reemplazar el conjunto de terminales $T = \{x_{t-1}, \dots, x_{t-p}\} \cup \{w_1, \dots, w_{l^*}\} = \{\mathbf{x} \cup \mathbf{w}\}$ única y exclusivamente por bloques funcionales, de acuerdo con la expresión: $T = \{B_1(\mathbf{x}, \mathbf{w}^{(1)}), \dots, B_l(\mathbf{x}, \mathbf{w}^{(l)})\}$, donde $\boldsymbol{\theta} = [\boldsymbol{\theta}^{(e)}, \mathbf{w}^{(1)}, \dots, \mathbf{w}^{(l)}]$. Donde, $\boldsymbol{\theta}^{(e)}$ corresponde al vector de parámetros generales, $\mathbf{w}^{(i)}$ corresponde al vector de coeficientes del bloque funcional $B_i(.)$. Además, cada bloque funcional es evaluado sobre todo el vector de rezagos $\mathbf{x}$, reemplazando los coeficientes w_j por los parámetros optimizados contenidos en el vector $\mathbf{w}^{(i)}$ para cada bloque funcional B_i.

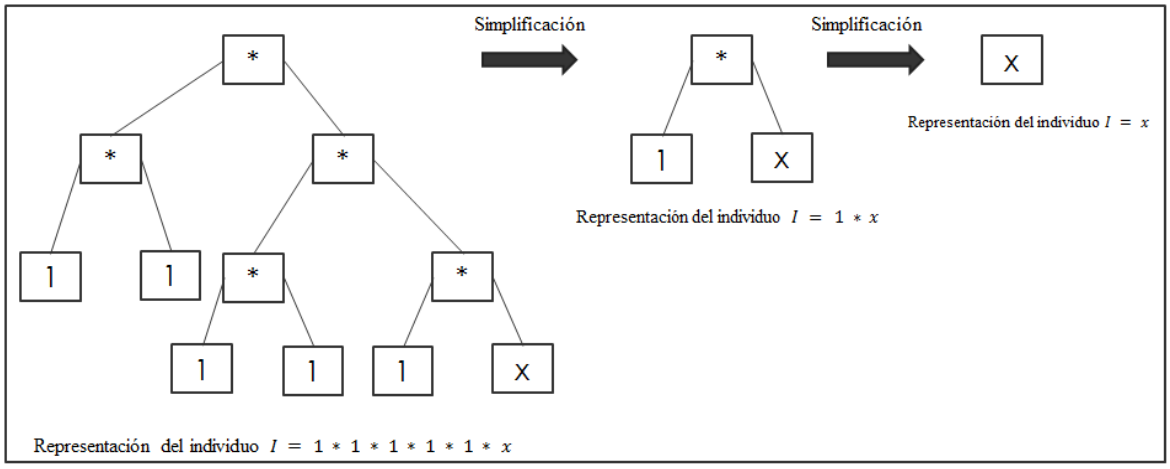


Figura 2-3: Ejemplo de simplificación de las constantes de los individuos.

Actualmente el espacio de búsqueda del algoritmo de GP, corresponde a un espacio funcional tradicional (A) formado por todas las posibles combinaciones de terminales y operadores, tal que:

$$A = \{T \circledast Z\} = \left\{\left\{\{x_{t-1}, \dots, x_{t-p}\} \cup \{w_1, \dots, w_l\}\right\} \otimes Z \otimes \dots \otimes \left\{\{x_{t-1}, \dots, x_{t-p}\} \cup \{w_1, \dots, w_l\}\right\} \dots\right\}$$

donde, Z corresponde al conjunto de operadores, $T = \left\{\{x_{t-1}, \dots, x_{t-p}\} \cup \{w_1, \dots, w_{l^*}\}\right\}$ equivale al conjunto de terminales, $\circledast$ es un operador que genera todas las posibles combinaciones entre los elementos de los dos vectores Z y T tal que la combinación es de tamaño infinito al permitirse el uso de los elementos de cada vector más de una vez, sin embargo solo son permitidas las combinaciones entre elementos $T_{1\leq i\leq l}$ del vector T a través de al menos un elemento $\zeta_{1\leq i\leq k}$ del vector Z; además, no es posible combinaciones de solo elementos consecutivos del vector Z, es decir: $\{Z \circledast T\} \cup \{\zeta_1, \dots, \zeta_l, \dots, \zeta_i \dots \zeta_j\} = \emptyset$. Por lo que: $\{Z \circledast T\} = \{T_1, \dots, T_l, T_1\zeta_1T_1, T_1\zeta_1T_2, \dots, T_2\zeta_1T_1, \dots, T_1\zeta_2T_1, \dots, T_1\zeta_1T_1\zeta_2T_1, \dots, T_1\zeta_1T_2\zeta_2T_1, \dots, T_i\zeta_j \dots T_a, \dots\}$. Asi mismo, una interpretación del operador $\circledast$ corresponde al uso iterativo del operador $\otimes$, el cual realiza todas las posibles combinaciones (finitas) entre los elementos de dos conjuntos de tamaño finito. Lo anterior implica que A considera todas las posibles combinaciones desagregadas de operadores y terminales, por lo que A constituye un espacio funcional de tamaño infinito ($dim(A) = \infty$).

Al utilizar exclusivamente bloques funcionales como elementos del conjunto de terminales T, se genera una reducción del espacio de búsqueda, al considerar únicamente aquellas combinaciones de subárboles (equivalente a un bloque funcional) de interés, agrupadas en un único nodo terminal. Dichos bloques funionales son seleccionados a partir de los modelos más utilizados en la literatura de predicción de series de tiempo, lo que implica que la combinación lineal/no lineal de los mismos, constituirá una hibridación de los modelos existentes. De acuerdo a lo anterior, el espacio de búsqueda del algoritmo de GP se transforma en: $A' = \{T \circledast Z\} = \left\{\left\{B_1\left(\mathbf{x}, \mathbf{w}^{(1)}\right), \dots, B_l\left(\mathbf{x}, \mathbf{w}^{(l)}\right)\right\} \otimes Z \otimes \dots \otimes \left\{B_1\left(\mathbf{x}, \mathbf{w}^{(1)}\right), \dots, B_l\left(\mathbf{x}, \mathbf{w}^{(l)}\right)\right\} \dots\right\}$, el cual es un espacio equivalente de tamaño infinito ($dim(A') = \infty$), pero es más pequeño que el espacio funcional tradicional A, debido a que el $\dim(A') \ll \dim(A)$, dado que existen combinaciones (arboles/individuos) pertenecientes a A que no están contenidos en A', mientras que todo individuo de A' esta contenido en A, lo cual cumple que: $\{\{\forall a' \in A' \; \exists a \in A \mid a' \equiv a\} \wedge \{\exists a \in A \mid \forall a' \in A' \; a' <> a\}\}$.

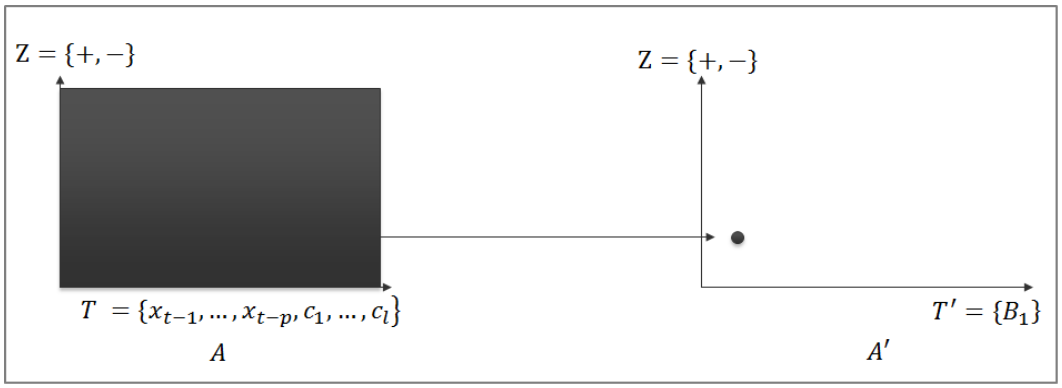


Figura 2-4: Ejemplo de cambio de espacio de búsqueda polinomial considerando únicamente bloques funcionales como terminales de los individuos de GP.

Por ejemplo, en la Figura 2-4 se muestra de forma gráfica el cambio en el espacio de búsqueda entre el espacio funcional tradicional A y el espacio funcional propuesto A', para el caso particular en el que el espacio funcional A esta formado por todas las combinaciones entre el conjunto de operadores $Z = \{+, -\}$ y terminales

$T = \{x_{t-1}, \dots, x_{t-p}, c_1, \dots, c_l\}$, equivalente a combinaciones lineales entre las constantes y rezagos definidos en T. Y el espacio funcional propuesto en este trabajo A', formado por todas las combinaciones entre el conjunto de operadores $Z = \{+, -\}$ y terminales $T' = \{B_1\}$. De acuerdo con lo anterior, todos los posibles individuos $I_a \in A$ pueden ser representados por un único individuo $I_{a'} = B_1 \in A'$, debido a que cualquier individuo $I_a \in A$ es una combinación lineal de los elementos de T, igual a: $\{\forall I_a \in A, I_a = c_1 + \cdots + c_1 + c_2 + \cdots + c_2 + c_l \dots . + c_l + x_{t-1} + \cdots + x_{t-1} + x_{t-2} + \cdots + x_{t-p} + \cdots + x_{t-p}\}$. Y dado que $B_1 = \sum_{i=1}^{p} w_i x_{t-i}$, es equivalente a la combinación lineal de todos los rezagos considerados con mejor valor de aptitud por medio de la optimización de sus parametros w_i, los cuales son el resultado de aplicar un algoritmo de optimización de parámetros (para este trabajo identificado como $Opt(.)$, ver Subsección 2.2.2); es posible afirmar que todo el espacio funcional de búsqueda del algoritmo de GP (A) puede ser representado por un único punto en A', donde el valor de aptitud B_1es mayor o igual al valor de aptitud de cualquier elemento de A (B_1 posee un error de predicción menor o igual que cualquier elemento de A).

De acuerdo con la ecuación 2.3, los bloques funcionales utilizan el vector de parámetros $\boldsymbol{\theta}$ y todo el vector de rezagos **x**, lo cual permite garantizar la inclusión de todos aquellos rezagos que sean de interés (definidos) para el modelo de predicción de la serie de tiempo analizada; mientras que los parámetros **w**, definen el real aporte de cada rezago a la disminución de la medida de error (aumento del valor de aptitud).

Algoritmo 2.2. Generación de individuos con bloques funcionales.

```
01: H_max
02  ϰ_max
03  T ← {B_1, ..., B_l}
04  Z ← {ζ_1, ..., ζ_k}
05  I ← { }
06  H ← 0
07  ϰ ← 0
08  loop while H ≤ H_max and ϰ ≤ ϰ_max and {∃j ∈ ℕ⁺ | Node(I,j) ⊄ T}
09        a ← random(0,1)
10        if a ≤ 0.5 then
11              Node(I,j) ← T_ceil(random(1,p+l))
12        else
13              Node(I,j) ← Z_ceil(random(1,k))
15              for k =1 to numArg(Node(I,j)) do
16                    Node(I,j+k) ← { }
17              end for
18              H ← H + 1
19        end if
20        ϰ ← ϰ + 1
21: end loop
22. I(x,θ) ← I(x,Opt(I))
23: end algorithm
```

Debido a que el algoritmo de GP realiza una exploración en el espacio de búsqueda definido por todas las posibles combinaciones entre los operadores y terminales seleccionados, y dado que se propone el uso exclusivo de terminales tipo bloque funcional, la búsqueda se reduce a encontrar aquellas formas funcionales que realmente aporten a la mejora de la aptitud del individuo, debido a que los parámetros equivalen al marco de referencia de la forma funcional definida, son definidos por medio de la aplicación del algoritmo de optimización $Opt(.)$. Por ejemplo, en la Figura 2-5 se pueden apreciar dos individuos I_1 y I_2 (funciones) cuya estructura funcional es distinta: I_1 corresponde a una línea recta cuya función de generación es $y = 2x + 2$; mientras, I_2 corresponde a una circunferencia con ecuación $x^2 + y^2 = 1$. Es posible apreciar que la estructura

funcional de las ecuaciones determina la respectiva forma del individuo, mientras que los parámetros definen la pendiente, la amplitud (radio) y el desplazamiento de los mismos (marco de referencia y perturbación).

De acuerdo a lo anterior, es necesario modificar el procedimiento de generación de los individuos presentado en el Algoritmo 2.1, adicionando un paso adicional (línea 22) de optimización de los parámetros $\boldsymbol{\theta}$ del individuo, el cual es presentado en el Algoritmo 2.2.

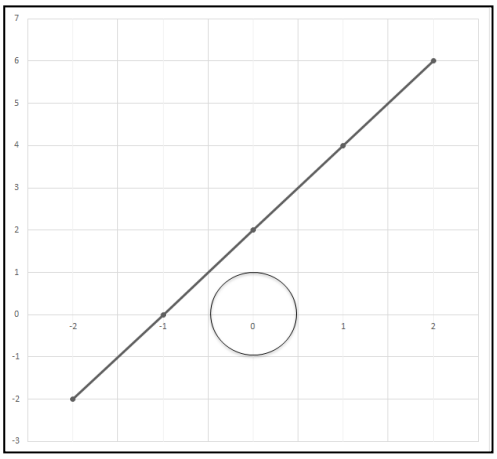


Figura 2-5: Representación de los individuos $I_1 = \ y = 2x + 2$ y $I_2 = x^2 + y^2 = 1$ en $\mathbb{R}^2$.

En conclusión, el Algoritmo 2.2 muestra la implementación de los conceptos —propuestos en esta tesis— de reducción del espacio de búsqueda del algoritmo de GP por medio de la eliminación de los terminales tipo constante, además de la inclusión de todos los rezagos considerados durante el proceso de generación de los individuos, lo cual garantiza, soportado por el uso del algoritmo de optimización de parámetros $Opt(.)$, que el valor de aptitud esta más relacionado con la forma funcional del individuo que por la inclusión de un rezago especifico.

2.1.3 Agrupación y cambio en la selección de los bloques funcionales

En el algoritmo original de GP, se toma un conjunto de operadores Z y terminales T para la generación aleatoria de los individuos. El vector de terminales T, lo constituye los distintos rezagos y constantes a ser utilizadas como hojas del árbol sintáctico que representa el individuo I, mientras que los operadores suele ser el conjunto de operadores aritméticos básicos $\{+, -, *, /\}$.

En la Subsección 2.1.2 se propone el reemplazo de los terminales por un conjunto de solo bloques funcionales $T = \{B_1(\mathbf{x}, \boldsymbol{\theta}), \dots, B_l(\mathbf{x}, \boldsymbol{\theta})\}$; sin embargo, teniendo en cuenta que en la Subsección 2.1.1 se muestra que el individuo I equivale a una aproximación de una serie de tiempo, y que las series de tiempo son expresadas en función de sus componentes de ciclo, tendencia, estacionalidad y error $\{S(.), C(.), \Gamma(.), \Upsilon(.)\}$ (ecuación 2.1); es necesario agrupar el conjunto de terminales T en subconjuntos de acuerdo con su componente especifica, con el fin de mantener la coherencia matemática del modelo resultante. De acuerdo a lo anterior, en este trabajo se propone reemplazar el conjunto de terminales T propuesto en la Subsección 2.1.3 por:

$$T = \begin{Bmatrix} S = \{B_1(\mathbf{x}, \boldsymbol{\theta}), \dots, B_a(\mathbf{x}, \boldsymbol{\theta})\}, \\ C = \{B_{\mathrm{a+1}}(\mathbf{x}, \boldsymbol{\theta}), \dots, B_{a+b}(\mathbf{x}, \boldsymbol{\theta})\}, \\ \Gamma = \{B_{\mathrm{a+b+1}}(\mathbf{x}, \boldsymbol{\theta}), \dots, B_{a+b+u}(\mathbf{x}, \boldsymbol{\theta})\}, \\ \Upsilon = \{B_{\mathrm{a+b+u+1}}(\mathbf{x}, \boldsymbol{\theta}), \dots, B_{a+b+u+v}(\mathbf{x}, \boldsymbol{\theta})\} \end{Bmatrix} \quad (2.10)$$

Algoritmo 2.3. Generación de individuos por medio del cálculo de λ.

H_{max}
$\varkappa_{max}$
ς
x
$S \leftarrow \{B_1, \dots, B_a\}$
$C \leftarrow \{B_{a+1}, \dots, B_{a+b}\}$
$\Gamma \leftarrow \{B_{a+b+1}, \dots, B_{a+b+u}\}$
$\Upsilon \leftarrow \{B_{a+b+u+1}, \dots, B_{a+b+u+v}\}$
$T \leftarrow \{S, C, \Gamma, \Upsilon\}$
$\mathrm{Z} \leftarrow \{\zeta_1, \dots, \zeta_k\}$
$I \leftarrow \{\ \}$
$H \leftarrow 0$
$\varkappa \leftarrow 0$
$\boldsymbol{\lambda} \leftarrow [\lambda_1, \lambda_2, \lambda_3, \lambda_4] \leftarrow statA(X)$
$s\lambda \leftarrow \sum_{\forall i | \lambda_i > 0} \lambda_i + \sum_{\forall i | \lambda_i \leq 0} \varsigma$
$\boldsymbol{\beta} \leftarrow \lambda / s\lambda$
loop while $H \leq H_{max}$ **and** $\varkappa \leq \varkappa_{max}$ **and** $\{\exists j \in \mathbb{N}^+ \mid Node(I,j) \not\subseteq T\}$
    $a \leftarrow random(0,1)$
    if $a \leq 0.5$ **then**
        $a \leftarrow random(0,1)$
        if $a \leq \beta_1$ **then**
            $Node(I,j) \leftarrow T_{ceil(random(1,a))}$
        elseif $a \leq \beta_1 + \beta_2$ **then**
            $Node(I,j) \leftarrow T_{ceil(random(a+1,a+b))}$
        elseif $a \leq \beta_1 + \beta_2 + \beta_3$ **then**
            $Node(I,j) \leftarrow T_{ceil(random(a+b+1,a+b+u))}$
        else
            $Node(I,j) \leftarrow T_{ceil(random(a+b+u+1,a+b+u+v))}$
        end if
    else
        $Node(I,j) \leftarrow \mathrm{Z}_{ceil(random(1,k))}$
        for k =1 **to** $numArg(Node(I,j))$ **do**
            $Node(I, j+k) \leftarrow \{\ \}$
        end for
        $H \leftarrow H + 1$
    end if
    $\beta \leftarrow validateMath(I, \boldsymbol{\beta})$
    $\varkappa \leftarrow \varkappa + 1$
end loop
$I(\mathbf{x}, \boldsymbol{\theta}) \leftarrow I(\mathbf{x}, Opt(I))$
end algorithm

La importancia de la división de los terminales en las respectivas componentes de la serie temporal, radica en la posibilidad de mezclar bloques funcionales (funciones) violando reglas matemáticas y sin interpretación directa en la realidad.

Adicionalmente, la división de terminales permite el enfoque del proceso de búsqueda del algoritmo de GP en zonas de poco interés, debido a las propiedades estadísticas de la serie temporal (problemas que se presentan en la implementación tradicional de GP y en la versión propuesta en la tesis de maestría).

Es de anotar, que cada serie de tiempo posee características estadísticas particulares que permiten identificar, con cierta certeza, la presencia de las componentes de estacionalidad, ciclo, tendencia y error. Por

lo que para la selección de los terminales y la asignación inicial de la probabilidad de selección de ellos, debe estar en concordancia con dichos análisis.

Por lo anterior, se propone realizar un análisis estadístico previo de la serie de tiempo (por ejemplo, el análisis por medio del auto correlograma parcial no lineal), definiendo los componentes presentes en la serie de tiempo en un vector $\boldsymbol{\lambda} = \{\lambda_1, \dots, \lambda_4\}$ dicotómico [0,1]. Luego, calcular el vector $\boldsymbol{\beta}$ de probabilidad de selección de la componente especifica de la serie de tiempo, por medio de la expresión:

$$\beta_i = \begin{cases} \lambda_i \; si \; \forall_j \lambda_j > 0 \\ \lambda_i / \sum \left(\begin{cases} \lambda_j \; si \; \lambda_j > 0 \\ \varsigma \; en \; otro \; caso \end{cases} \right) \; en \; otro \; caso \end{cases} \tag{2.11}$$

Es de resaltar, que definir un $\varsigma \in \mathbb{R}$ pequeño (y no estrictamente 0), garantiza que cada grupo de bloques funcionales tengan alguna probabilidad de selección, con el fin de identificar estructuras funcionales de interés que no cumplen los criterios de combinación entre componentes de la serie temporal durante la generación de los individuos en el proceso de búsqueda del algoritmo de GP; debido a información faltante o componentes de error externo que distorsionen las características estadisticas de la serie temporal.

Para implementar la ecuación 2.11 es necesario modificar el procedimiento de generación de los individuos presentado en el Algoritmo 2.2, adicionando los pasos para el cálculo de $\boldsymbol{\lambda}$ y la probabilidad de selección de los bloques funcionales asociados a una componente específica $\boldsymbol{\beta}$, la cual es presentada en el Algoritmo 2.3.

En el Algoritmo 2.3 se introduce la función $statA(\mathbf{x})$, la cual evalúa las propiedades estadísticas de la serie de tiempo representada por **x,** generando un el vector dicotómico $\boldsymbol{\lambda} = \{\lambda_1, \dots, \lambda_4\}$. La función $validateMath(I, \boldsymbol{\beta})$ analiza las componentes actuales del individuo I y actualiza el vector $\boldsymbol{\beta} = \{\beta_1, \dots, \beta_4\}$ para garantizar consistencia matemática.

La función $validateMath(I, \boldsymbol{\beta})$ permite identificar las principales componentes presentes en la serie de tiempo por medio de los análisis presentes en la literatura de predicción de series de tiempo:

- **Número de rezagos a ser considerados**
  El número de rezagos a ser considerados son el resultado del análisis del autocorrelograma parcial entre los rezagos definido por Nielsen & Madsen [110]: $PLDF(i) = sgn\big(f_{ii}(b) - f_{ii}(a)\big)\sqrt{R^2_{(0i)(1,..i-1)}}$, donde a y b son los valores mínimo y máximo sobre las observaciones. $f(.)$, y $R^2_{(0i)(1,..i-1)} = \frac{SS_{0(1,\dots,i-1)} - SS_{0(1,\dots,i)}}{SS_{0(1,\dots,i-1)}}$. Sin embargo debido a la complejidad de calculo se opto por la interpretación de PLDF a la luz de las redes neuronales artificiales propuesto por Martínez [111].

- **Componente de tendencia**
  Se aplica la prueba de Mann-Kendall, propuesto por Mann [112] y estudiado por Kendall [113], en el cual se analiza la hipótesis H_0: La serie presenta componente de tendencia.

- **Componente de estacionalidad**
  Se aplica la prueba de Mann-Kendall [112, 113] en el cual se analiza si existe la raíz de un modelo autoregresivo $x_t = px_{t-1} + u_t$ donde donde x_{t-1} corresponde al rezago en $t-1$, y u_t es el error asociado. Por lo que si existe la raíz en $p = 1$ el modelo sería no estacionario en este caso. Por lo que se suele utilizar el análisis de sus diferencias dados por la expresión: $\Delta x_t = (p-1)x_{t-1} + u_t$, en el cual se analiza la prueba con los datos residuales por medio de la distribución T asociada, llamada tabla Dickey-Fuller [114].

- **Componente de Ciclo**
  Para evaluar la componente de ciclo en la serie de tiempo se opto por utilizar el test de Priestley-Subba Rao (PSR) [115, 116], en la cual $Y(t,w) = \log(Ft(w))$, donde $Ft(w)$ es un estimado de f, con aproximación: $E[Y(t,w)] = \log(ft(w))$, y la varianza de $Y(t,w)$ es aproximadamente una constante, siendo el logaritmo el estabilizador de la varianza y concentrando la evaluación en el cambio del promedio de Y.

2.2 Modificaciones al algoritmo de programación genética

Para el logro de los objetivos específicos:

- *Proponer un nuevo mecanismo de generación de individuos de programación genética por medio de agrupaciones de bloques funcionales, con el fin de garantizar individuos lógicos de manera matemática, y su posterior interpretación a la luz de las variables propias de modelación.*
- *Modificar el proceso de generación de los individuos de programación genética para garantizar la inclusión de cada uno de los rezagos de interés, con el fin de disminuir la influencia de los terminales en la selección de la estructura funcional de los individuos.*

es necesaria la modificación de las distintas componentes del algoritmo de GP, esto incluye la generación de los individuos, la aplicación de los operadores de reproducción, cruce y selección, adicional a la definición de los operadores de mutación, simplificación e intensificación. Por lo que, en esta sección se realiza un análisis de las componentes del algoritmo de GP, las modificaciones propuestas y su relevancia para el logro de los objetivos descritos.

2.2.1 Modificación de la generación de la población inicial

Teniendo en cuenta que en la Subsección 2.1.2 se plantea el uso exclusivo de bloques funcionales como terminales de los individuos, y en la Subsección 2.1.3 se plantea la agrupación de los bloques funcionales en componentes de estacionalidad, ciclo, tendencia y error de acuerdo con la ecuación 2.10, los cuales son definidos a partir de un análisis estadístico previo de la serie temporal; es necesario replantear el procedimiento de generación de la población inicial P_0 del algoritmo original de GP, el cual corresponde al Algoritmo 2.4, en el cual, a partir de los vectores T y Z, el tamaño de la población n, el nivel de profundidad máximo inicial por individuo $\varkappa_{max}$ y el número máximo de nodos H_{max}, es generado de manera aleatoria el árbol sintáctico equivalente al individuo. Por lo que el Algoritmo 2.4 es equivalente a ejecutar n veces el Algoritmo 2.1, almacenando el resultado en un vector P_0 equivalente a la población inicial de individuos.

En el Algoritmo 2.5 se muestra el procedimiento propuesto de generación de la población inicial, de acuerdo con el procedimiento del Algoritmo 2.3, en el que se incluye los pasos de cálculo de $\boldsymbol{\lambda}$ y la probabilidad de selección de los bloques funcionales asociados a una componente específicos $\boldsymbol{\beta}$, para el cálculo de cada uno de los individuos que componen a $P_0 = \{I_1, \dots, I_n\}$.

En el algoritmo tradicional de GP y la propuesta realizada en la tesis de maestría [12], la probabilidad de selección de los terminales es proporcional al número de los mismos $prb = 1/l$, sin considerar que las propiedades estadísticas de la serie de tiempo permiten seleccionar estructuras funcionales más acordes a partir del análisis del vector de entradas $\mathbf{x}$. Por lo anterior, se propone el uso de un vector de probabilidades de selección por componente ($\{S, C, \Gamma, \Upsilon\}$) que permita guiar los individuos del algoritmo de GP durante el proceso de búsqueda hacia zonas de interés.

Por lo anterior, en el Algoritmo 2.5 se incluye un vector de probabilidades de selección por bloque funcional $prbT$ por medio de la instrucción $prbT \leftarrow \{calculatePrb(\mathbf{x}, S),\ calculatePrb(\mathbf{x}, C),\ calculatePrb(\mathbf{x}, \Gamma),\ calculatePrb(\mathbf{x}, \Upsilon)\}$, en la cual la función $calculatePrb(\mathbf{x}, .)$ genera un vector de probabilidades prb tal que $1 = \sum prbT_i$ a partir del análisis de los datos **x** de la serie de tiempo, para la componente especifica. La probabilidad de selección proporcional utilizada en el algoritmo tradicional de GP (y propuestas presentes en la literatura) equivaldría a $calculatePrb(.) = \{1/size(a), \ldots.,1/size(a)\}_{size(a)}$.

Otra alternativa de introducción de la probabilidad de selección no proporcional en las componentes corresponde a la replicación de $Ceil(Prb_k * 100)$ veces el bloque funcional k de acuerdo con el vector $prbT$. Al utilizar esta propuesta, es posible mantener la probabilidad de selección para cada uno de los bloques funcionales intrínsecamente en el vector general de terminales T, sin modificar el método de selección del terminal de acuerdo con el algoritmo original de GP. Es asi que, la componente específica $T_i \in \{S, C, \Gamma, \Upsilon\}$ de T puede ser calculada como: $T_i = \left\{B_{1,1}, \ldots, B_{1,ceil(Prb_1*100)}, \ldots, B_{size(T_i),1}, \ldots, B_{size(T_i),ceil(Prb_{size(T_i)}*100)}\right\}$. Aumentando la probabilidad a ciertos bloques funcionales de ser seleccionados en los individuos generados aleatoriamente, y asi las soluciones (individuos) se concentrarán en estructuras funcionales ubicadas en zonas de interés definidas.

Algoritmo 2.4. Generación de población inicial – Algoritmo de original de GP.

```
01  H_max
02  ϰ_max
03  n
04  P_0 = { }
05  T ← {x_{t-1}, ..., x_{t-p}} ∪ {α_1, ..., α_{l*}}
06  Z ← {ζ_1, ..., ζ_k}
07  for i = 1 to n do
08          H ← 0
09          ϰ ← 0
10          I ← { }
11          loop while H ≤ H_max and ϰ ≤ ϰ_max and {∃j ∈ ℕ+ | Node(I,j) ⊄ T}
12                  a ← random(0,1)
13                  if a ≤ 0.5 then
14                          Node(I,j) ← T_{ceil(random(1,p+l))}
15                  else
16                          Node(I,j) ← Z_{ceil(random(1,k))}
17                          for k =1 to numArg(Node(I,j)) do
18                                  Node(I,j+k) ← { }
19                          end for
20                          H ← H + 1
21                  end if
22                  ϰ ← ϰ + 1
23          end loop
24          P_0 = P_0 ∪ I
25  end for
26  end algorithm
```

Dado que se optó por la selección de bloques funcionales por componente a partir de un vector de probabilidades $prbT$, en el método sin replicación de bloques funcionales, debido principalmente a la precisión en la selección de un bloque funcional especifico, es necesario adicionar la función $selBF(a, b, prbT)$, la cual genera el índice del bloque aleatorio a ser seleccionado entre a y b a partir del vector $prbT$, cuyo funcionamiento se puede apreciar en el Algoritmo 2.6.

Algoritmo 2.5. Generación de la población de inicial – propuesta con BF.

H_{max}
$\varkappa_{max}$
ς
x
n
$P_0 = \{\}$
$S \leftarrow \{B_1, \dots, B_a\}$
$C \leftarrow \{B_{a+1}, \dots, B_{a+b}\}$
$\Gamma \leftarrow \{B_{a+b+1}, \dots, B_{a+b+u}\}$
$\Upsilon \leftarrow \{B_{a+b+u+1}, \dots, B_{a+b+u+v}\}$
$T \leftarrow \{S, C, \Gamma, \Upsilon\}$
$prbT \leftarrow \{calculatePrb(\mathbf{x}, S), calculatePrb(\mathbf{x}, C), calculatePrb(\mathbf{x}, \Gamma), calculatePrb(\mathbf{x}, \Upsilon)\}$
$\mathrm{Z} \leftarrow \{\zeta_1, \dots, \zeta_k\}$
$\boldsymbol{\lambda} \leftarrow [\lambda_1, \lambda_2, \lambda_3, \lambda_4] \leftarrow statA(X)$
$s\lambda \leftarrow \sum_{\forall i | \lambda_i > 0} \lambda_i + \sum_{\forall i | \lambda_i \leq 0} \varsigma$
for $i = 1$ **to** n **do**
    $I \leftarrow \{\}$
    $H \leftarrow 0$
    $\varkappa \leftarrow 0$
    $\boldsymbol{\beta} \leftarrow \lambda / s\lambda$
    loop while $H \leq H_{max}$ **and** $\varkappa \leq \varkappa_{max}$ **and** $\{\exists j \in \mathbb{N}^+ \mid Node(I, j) \notin T\}$
        $a \leftarrow random(0,1)$
        if $a \leq 0.5$ **then**
            $a \leftarrow random(0,1)$
            if $a \leq \beta_1$ **then**
                $Node(I, j) \leftarrow T_{selBF(1,a,prbT)}$
            elseif $a \leq \beta_1 + \beta_2$ **then**
                $Node(I, j) \leftarrow T_{selBF(a+1,a+b,prbT)}$
            elseif $a \leq \beta_1 + \beta_2 + \beta_3$ **then**
                $Node(I, j) \leftarrow T_{selBF(a+b+1,a+b+u,prbT)}$
            else
                $Node(I, j) \leftarrow T_{selBF(a+b+u+1,a+b+u+v,prbT)}$
            end if
        else
            $Node(I, j) \leftarrow Z_{ceil(random(1,k))}$
            for k =1 **to** $numArg(Node(I, j))$ **do**
                $Node(I, j + k) \leftarrow \{\}$
            end for
            $H \leftarrow H + 1$
        end if
        $\boldsymbol{\beta} \leftarrow validateMath(I, \beta)$
        $\varkappa \leftarrow \varkappa + 1$
    end loop
    $I(\mathbf{x}, \Theta) \leftarrow I(\mathbf{x}, Opt(I))$
    $P_0 = P_0 \cup I$
end for
end algorithm

En general, en el Algoritmo 2.5 se desarrollan los siguientes pasos:

1. Análisis de las propiedades estadísticas de la serie de datos.
2. Selección del grupo de terminales y operadores de acuerdo al Punto 1.

3. Distribuir la probabilidad de selección de los bloques funcionales en cada uno de los grupos de terminales de acuerdo con información de la serie.
4. Cada uno de los bloques funcionales tendrá una probabilidad ($prbT_i$) de selección, la cual influenciará su selección en el proceso de generación (se puede reemplazar como la copia consecutiva de los mismos en múltiplos enteros de la prb).
5. Generar aleatoriamente el individuo de acuerdo a $H_{max}, \varkappa_{max}, Z$, el grupo de terminales seleccionados en 2 y 3, y el vector de probabilidades $prbT$.

Los cambios propuestos permiten la focalización de los individuos en estructuras funcionales específicas de interés, equivalentes a los bloques funcionales, logrando la disminución del espacio de búsqueda y aumento la probabilidad de convergencia a estructuras de mayor aporte al valor de aptitud del individuo.

Algoritmo 2.6. Selección del índice del bloque funcional a ser utilizado.

```
01 Function selBF(a, b, prbT)
02         r ← random(0,1)
03         i ← a
04         d ← prbT_i
05         loop while d ≤ r
06                 i ← i + 1
07                 d ← d + prbT_i
08         end loop
09         return i
10 end function
11 end algorithm
```

2.2.2 Modificación del cálculo de la aptitud

En la propuesta de modificación de los terminales T de esta tesis, se define el uso único y exclusivo de los bloques funcionales, y que los parámetros $\boldsymbol{\theta}$ sean optimizados en cada cambio estructural del individuo, por lo que es necesario la optimización del vector $\boldsymbol{\theta}$ de cada individuo I_i de la población actual P_g por medio de la función $Opt(.)$ durante todo el proceso de búsqueda del algoritmo de GP.

Una vez optimizado el individuo I_i, es necesario evaluar que tan "bueno" es el individuo a partir de la función de aptitud $Opt(I_i, \dots)$ seleccionada, tradicionalmente se suele utilizar una medida de error basada en distancia, como son: la suma de los errores absolutos (ASE, por su sigla en inglés), suma del error cuadrático (SSE por su sigla en inglés), y el promedio de la suma del error cuadrático (MSE, por su sigla en inglés). Sin embargo, una función de aptitud $Apt(I_i)$ basada únicamente en la medida de error de aproximación entre $\hat{F}(.) = I_i(\mathbf{x}, \boldsymbol{\theta})$ y $F(.)$ genera un crecimiento desmedido de los individuos (número de nodos), sin un aporte real a la disminución del error, lo cual genera un detrimento en la calidad de las soluciones encontradas durante el proceso de búsqueda [81, 82], lo cual suele ser llamado bloat.

En la literatura actual de GP (Tabla 1-1) se han realizado intentos por disminuir el bloat por medio del uso del algoritmo de limitación dinámica de profundidad (DDL) [11]; el uso de métodos de simplificación de rango y la inclusión del operador de simplificación de regresión lineal [10]; el uso del método de simplificación de decisión equivalente (EDS) [62]; la codificación binaria de los individuos y posterior reemplazo por únicos bits [63]; la simplificación algebraica a partir de reglas aritméticas [64]; el uso de nodos de alta densidad (HDN) para el almacenamiento de información adicional en los nodos de los individuos de GEP [65]; la evaluación de la función de aptitud de los subárboles internos de los individuos para su selección como nuevos individuos de la población de GP [67]; la eliminación de terminales irrelevantes por medio de la evaluación de su aporte a la aptitud total del individuo [11]; el uso de evolución diferencial (DE) para la determinación de las constantes a ser utilizadas por los individuos [66]; el uso de módulos emergentes desacoplados persistentes

entre generaciones para ser utilizados en la generación de nuevos individuos [68]; la redefinición de los individuos de GP por medio de bloques poligonales [7]; y las modificaciones propias para el uso de operadores de cruce de más de dos individuos [61]. Sin embargo, dichos métodos están basados principalmente en la eliminación de nodos y simplificación de los mismos en el árbol correspondiente al individuo I_i, sin abordar el problema de la medida de error utilizada, siendo la medida de error la base fundamental de selección de los individuos que prevalecerán durante el proceso de búsqueda.

En esta Subsección se analizarán las medidas de complejidad presentes en la literatura de programación genética, adicional, a proponer la medida de aptitud a ser utilizada.

2.2.3 Medidas de complejidad utilizadas en programación genética

Tradicionalmente se han utilizado medidas de error de predicción basadas en la distancia entre el valor real $F(\mathbf{x}, \boldsymbol{\theta})$ y el pronosticado $I_i(\mathbf{x}, \boldsymbol{\theta})$:

- *Suma del error cuadrático (SSE, por su sigla en inglés)*
  Este indicador realiza una sumatoria de las diferencias cuadráticas entre los resultados generados por el individuo y los valores reales registrados por la serie de tiempo, equivalente a la expresión:

$$Apt(I_i) = SSE(I_i) = \sum \left(I_i(\mathbf{x}, \boldsymbol{\theta}) - F(\mathbf{x}, \boldsymbol{\theta})\right)^2$$

- *Suma del error cuadrático medio (MSE, por su sigla en inglés)*
  Este indicador realiza un promedio de la sumatoria de las diferencias cuadráticas entre los resultados generados por el individuo y los valores reales registrados por la serie de tiempo, equivalente a la expresión:

$$Apt(I_i) = MSE(I_i) = \frac{\sum \left(I_i(\mathbf{x}, \boldsymbol{\theta}) - F(\mathbf{x}, \boldsymbol{\theta})\right)^2}{size(\mathbf{x})} = SSE(I_i)/size(\mathbf{x})$$

- *Suma del error absoluto (ASE, por sigla en inglés)*
  Este indicador realiza una sumatoria de las diferencias absolutas entre los resultados generados por el individuo y los valores reales registrados por la serie de tiempo, equivalente a la expresión:

$$Apt(I_i) = ASE(I_i) = \sum \|I_i(\mathbf{x}, \boldsymbol{\theta}) - F(\mathbf{x}, \boldsymbol{\theta})\|$$

Y medidas basadas en la variación:

- *Desviación estándar (STD, por su sigla en inglés)*

$$Apt(I_i) = STD(I_i) = \sqrt{\frac{\sum \left(I_i(\mathbf{x}, \boldsymbol{\theta}) - F(\mathbf{x}, \boldsymbol{\theta})\right)^2}{(size(\mathbf{x}) - 1)}}$$

- *Desviación media absoluta (MAD, por su sigla en inglés)*

$$Apt(I_i) = MAD(I_i) = median(\|I_i(\mathbf{x}, \boldsymbol{\theta}) - F(\mathbf{x}, \boldsymbol{\theta})\|)$$

Sin embargo, en los últimos años han surgido medidas basadas en criterios de información —apalancadas por los efectos nocivos asociados a la degradación de la calidad de los individuos/soluciones de la población por el uso de medidas de aptitud basadas meramente en funciones de error [81, 82]— trabajadas por Montaña et al. [34], cuya forma general es: $\varepsilon(f) = \varepsilon_n(f) + pen(h/n, \sigma^2)$ ó $\varepsilon(f) = \varepsilon_n(f) + pen(h, n)$, donde $\varepsilon_n(f)$ es la aproximación del error de la máquina de aprendizaje por medio de la muestra finita: $\{x_i, y_i\}_{i=1}^{n}$: $\varepsilon_n(f) = \frac{1}{n}\sum_{i=1}^{n} Q(x_i, f, y_i)$, n es el tamaño de la muestra, f es la función (individuo para GP), y_i es el valor real de la función $F(.)$, pen es el factor de penalización, σ es la desviación estándar, h es la medida de complejidad del modelo (para GP suele ser el número de parámetros libres del modelo f; o el tamaño en número de nodos del individuo). Se suele utilizar $Q(x_i, f, y_i)$ como la diferencia cuadrática entre el aproximador y el valor real, por lo que $\varepsilon_n(f)$ equivaldría al $MSE(I_i)$.

Entre las principales medidas basadas en criterios de información se tienen:

- *Criterio de información Akaike – AIC [117]*

$$\varepsilon(f) = \varepsilon_n(f) + 2 * \frac{h}{n} * \sigma^2$$

- *Criterio de información Bayesiano – BIC [118]*

$$\varepsilon(f) = \varepsilon_n(f) + \log(n) * \frac{h}{n} * \sigma^2$$

- *Criterio de minimización del riesgo estructural – SRM [119]*

$$\varepsilon(f) = \varepsilon_n(f) * \left(1 - \sqrt{\frac{h}{n} - \ln\frac{h}{n} + \frac{\ln n}{2n}}\right)^{-1}$$

2.2.4 Medida de aptitud a ser utilizada

Para solventar el problema de la correcta selección de los individuos incluyendo información de la disminución de la medida de error y el tamaño de la solución (individuo), en este trabajo se propone el uso del criterio de información Akaike como función de aptitud $Apt(.)$, adaptándolo al algoritmo de GP propuesto, en el cual se pondera tanto la complejidad del individuo como el error de aproximación.

En la modificación propuesta del algoritmo de GP presentada en la subsección 2.1.2, en la cual se utilizan única y exclusivamente bloques funcionales como terminales del individuo, la expresión para el criterio de información Akaike se puede expresar como:

$$Apt(I_i) = Akaike(I_i) = Ln\left(\frac{(I_i(\mathbf{x}, \boldsymbol{\theta}_i) - F(\mathbf{x}, \boldsymbol{\theta}_i))^2}{size(I_i)}\right) + \frac{size(I_i) + size(\boldsymbol{\theta}_i)}{size(I_i) - \text{size}(\boldsymbol{\theta}_i) - 2} \tag{2.12}$$

Con size($\boldsymbol{\theta}_i$) igual al número de parámetros utilizados en el individuo $I_i \in P_g$.

La ecuación 2.12 permite sopesar el error de aproximación y la complejidad del modelo resultante (individuo), focalizando a la población de individuos P_g en aquellas zonas que estructuralmente generen mejoras sustanciales en la medida de error, con un tamaño menor del individuo.

2.2.5 Selección del algoritmo de optimización de los individuos

Debido a que la propuesta de uso bloques funcionales vista en la Sección 2.1, en la cual el conjunto de terminales son reemplazados única y exclusivamente por bloques funcionales, y los parámetros son optimizados por medio de un algoritmo de optimización $Opt(.)$ seleccionado. Se puede apreciar que el proceso

de búsqueda del algoritmo de GP se basará en una búsqueda de estructuras funcionales que generen mejor valor de aptitud, lo cual implica que es necesario garantizar que dos estructuras funcionales similares posean un valor similar de aptitud.

Para que dos individuos $\{I_1, I_2\}$ posean un valor de aptitud similar, se debe cumplir que: $|Apt(I_1) - Apt(I_2)| \leq \delta$, con $\delta \in \mathbb{R}$ pequeño. Dado que $Apt(.)$ tiene en cuenta el tamaño del individuo y la medida de error, y que el algoritmo $Opt(.)$ optimiza los parámetros de I_i, existe una clara relación entre el algoritmo seleccionado y las medidas de error de los individuos.

Por lo anterior, se propone utilizar algoritmos de optimización que cumplan que: Dado un punto inicial igual $\boldsymbol{\theta}_0$, el $|I_1(\mathbf{x}, \mathrm{Opt}(\mathbf{x}, \boldsymbol{\theta}_1)) - I_2(\mathbf{x}, \mathrm{Opt}(\mathbf{x}, \boldsymbol{\theta}_2))| \leq \delta$, con $\delta \in \mathbb{R}$ pequeño, esto es: $\forall_{i=\{1,\dots,size(I_1)\}} Node(I_1, i) = Node(I_2, i)$ y $size(I_1) = size(I_2)$ Por lo que se sugiere el uso de:

- Algoritmos de optimización basados en gradientes, en los cuales se garantiza que dado un punto inicial $\boldsymbol{\theta}_{i_0}$ siempre convergirán al mismo óptimo local con igual nivel de error.
- Realizar simulación con un número grande de iteraciones que permita seleccionar aquellos parámetros con menor medida de error teniendo en cuenta que no es posible garantizar la condición de generación semejante de manera estricta.

2.2.6 Modificación del operador genético de reproducción

El operador genético de reproducción tiene como objetivo la concentración de la población en zonas con similar estructura funcional y parámetros, para lo cual se mantiene una copia idéntica del individuo entre generaciones. Su implementación se ha realizado tradicionalmente por medio de una variable aleatoria que permite escoger o no su aplicación al individuo seleccionado de la población, de acuerdo con la expresión: $I_i = I_i \leftrightarrow random(0,1) \leq prbRep$, siendo $prbRep$ la probabilidad de selección de los individuos para la aplicación del operador genético de reproducción, con valores recomendados de 0.1 [3], aunque depende directamente de la serie de tiempo y la homogeneidad de la zona de búsqueda.

En este trabajo se conserva la definición original del algoritmo de GP ($I_i^* = I_i$), adaptándolo a la nueva clase de individuos constituidos por bloques funcionales, generando por ende un nuevo árbol idéntico, con todos sus terminales T, operadores Z y parámetros $\boldsymbol{\theta}$ iguales. Sin embargo, en series de tiempo con componente de tendencia Γ muy bajo, las estructuras de los individuos suelen corresponder con estructuras funcionales planas, en las cuales, un porcentaje de utilización alto no contribuye a la diversificación de las soluciones y la exploración de nuevas zonas de interés; mientras que, en estructuras con pendientes pronunciadas, es necesario focalizar la zona de búsqueda en zonas de interés dadas por los individuos en generaciones anteriores.

De acuerdo a lo anterior, en este trabajo se propone la utilización de un porcentaje dinámico de reproducción, basado en índices de concentración de las soluciones en el espacio de búsqueda, evaluando la mejora sucesiva en el valor de aptitud de los individuos en las generaciones anteriores de acuerdo a la siguiente expresión:

$$prbRep(P_g) = min\left(\kappa_0 + \xi\left(P_g\right) * \tau, \kappa_1\right) \tag{2.13}$$

donde $\kappa_0 \in [0, 1]$ es una constate real que equivale a la probabilidad mínima de aplicación del operador de reproducción en cualquier generación g, $\kappa_1 \in [0, 1]$ es una constante real que corresponde a la máxima probabilidad de aplicación del operador de reproducción a ser aplicado en cualquier generación g, $\tau \in \mathbb{R}$ es el porcentaje de incremento máximo, $\xi(P_g)$ es la función que analiza el porcentaje de mejora ponderado del total de individuos de la población por medio de la expresión: $\xi\left(P_g\right) = \sum mejor(I_i, P_{g-1})\,/n$, con $mejor(I_i, P_{g-1}) = 1$ si $Apt(I_i) > Max\left(Apt\left(P_{g-1}\right)\right)$.

Ésta modificación propuesta, permite tanto la concentración de los individuos en regiones de interés en la superficie del espacio de búsqueda como su diversificación a nuevas zonas, teniendo en cuenta el desempeño de las poblaciones anteriores y el porcentaje de mejora en las soluciones encontradas, permitiendo un mejor equilibrio entre la diversificación y la concentración de soluciones durante la ejecución del algoritmo de GP.

2.2.7 Modificación del operador genético de mutación

El operador genético de mutación tiene como objetivo introducir pequeños cambios en la estructura de los individuos que permita la diversificación de la población en el espacio de búsqueda.

Tradicionalmente, el operador genético de mutación se basa en la selección aleatoria de un nodo del individuo reemplazándolo por otro seleccionado del conjunto de terminales y operadores $\{T\ U\ F\}$. Sin embargo, al analizar la estructura de los individuos, y considerando que corresponden a ecuaciones con dimensión, estructura y parámetros, no es claro a que corresponde un cambio pequeño y mucho menos como lograrlo, por ello, a continuación se proponen algunas alternativas de tratamiento analizadas en virtud de sus ventajas y desventajas.

- *Cambio en una única dimensión*

En la modificación propuesta al algoritmo de GP en la Sección 2.1, los individuos son compuestos exclusivamente por bloques funcionales, y éstos actúan sobre todas las dimensiones del mismo (conjunto de rezagos **x**); siendo $\theta_i \neq 0$ los parámetros que definen si el rezago especifico es de relevancia para el individuo, dado que, al cambiar un bloque funcional de un individuo I_i podrían variar todas sus dimensiones, por lo que, y de acuerdo con la ecuación 2.14, al cambiar un único parámetro θ_k del bloque funcional j del individuo I_i, equivaldría a modificar una única dimensión.

Al analizar las condiciones de cambio de un único parámetro k del individuo I_i, optimizando los parámetros del mismo y manteniendo k constante, se puede apreciar que en la práctica es computacionalmente costoso y de convergencia más lenta, debido principalmente a que el operador genético de mutación equivaldría a:

a) Seleccionar un terminal B_j del individuo I_i de manera aleatoria.
b) Seleccionar uno de los rezagos $x_{t-\mathrm{a}}$ de B_j de manera aleatoria.
c) Cambiar el parámetro asociado al rezago seleccionado θ_k, así:

$$\theta_k = \begin{cases} \text{Si } \theta_k \leq \varsigma => \theta_k = \tau, \text{con } \varsigma \text{ pequeño y } \tau > \varsigma. \\ \text{Si } \theta_k \geq \varsigma => \theta_k = 0, \text{con } \varsigma \text{ pequeño.} \end{cases} \tag{2.14}$$

De acuerdo ésto, agregar o quitar bloques funcionales no modifica la dimensión del mismo $dim(\mathbf{x}) = n = número\ de\ rezagos\ considerados$, y solo es posible modificarlo intrínsecamente al definir parámetros en cero manteniendolos en ese valor durante la aplicación del algoritmo de optimización $Opt(.)$ al individuo I_i, lo anterior también es similar a agregar un bloque funcional con todos los parámetros en 0 exceptuando θ_k.

Adicionalmente, esta propuesta a la luz del funcionamiento del algoritmo modificado de GP, no aporta información adicional, dado que restringe al individuo a un único rezago sin considerar los demás rezagos de interés, y dado que, los individuos son optimizados por medio de un algoritmo $Opt(.)$, los parámetros ya tienen inmersa la información del nivel de aporte de cada uno. Además, si consideramos que los bloques funcionales corresponden a funciones aplicadas sobre todos los rezagos, cualquier individuo con igual estructura funcional y con solo un cambio en el vector de parámetros, es un caso particular del mismo, lo que implica que no aporta información adicional útil para el proceso de búsqueda del algoritmo de GP.

Por lo anterior, si bien corresponde a una posibilidad de desarrollo del operador de mutación más acorde con la definición de pequeños cambios en el individuo (siendo los individuos ecuaciones con estructura funcional y parámetros), no se utilizará en la implementación de esta propuesta metodológica.

- *Adición de un subárbol*

Debido a que en la propuesta de modificación del algoritmo de GP de la Sección 2.1, los individuos están conformados por bloques funcionales y estos evalúan la totalidad de rezagos, el cambio en la estructura funcional en torno a las dimensiones del individuo puede ser generado a partir de la agregación de un subárbol adicional al individuo.

El subárbol adicional puede ser un bloque funcional único (árbol con solo un nodo) o todo un individuo completo (un árbol con más de un nodo). La generación del árbol puede ser aleatorio o partir de algún individuo existente en la población, adicional a que la agregación puede ser por medio del operador aritmético suma (+) o multiplicación (*), estas posibilidades son analizadas a continuación:

- *Cambio en un operador*
  Tradicionalmente en el algoritmo de GP se han utilizado el conjunto de operadores aritméticos $Z = \{+, -, *, /\}$ [3], y dado que en la propuesta de la Sección 2.1 los individuos son conformados exclusivamente por bloques funcionales, los cuales se optimizan por medio de $Opt(.)$, entonces, si se toman los operadores $\{+\}$ y $\{*\}$, el intercambio de los mismos pueden generar deformaciones en el espacio de búsqueda como se analizará en los siguientes párrafos.

  Pasar de un operador suma $\{+\}$ a un operador multiplicación $\{*\}$ implica que los polinomios aumenten de grado, deformando la superficie de $\mathbb{R}^p$. Por el contrario, al pasar del operador multiplicación $\{*\}$ al operador suma $\{+\}$, implica que los grados de los polinomios disminuyan, y que se pueda replicar la forma funcional (estructura) de ambos subárboles (derecho e izquierdo) restando información al modelo.

  Por lo anterior, esta propuesta constituye una alternativa de implementación viable del operador genético de mutación; sin embargo, pueden ser cambiadas varias dimensiones al tiempo de acuerdo a la estructura del individuo.

- *Cambio en un terminal*
  Al cambiar un terminal en la versión tradicional del algoritmo de GP se suele cambiar la estructura al incorporar algún rezago (puedo no haber sido usado previamente), o constante que suele usarse como parámetro de optimización del individuo.

  En la versión propuesta en la Sección 2.1, los terminales solo están constituidos por bloques funcionales, y cada uno de ellos se optimiza por medio de $Opt(.)$, por lo que los parámetros $\boldsymbol{\theta}$ esta inmersos en el individuo, e implica que dos terminales con similar estructura funcional serán iguales ($|I_1(\mathbf{x}, \mathrm{Opt}(\mathbf{x}, \boldsymbol{\theta}_1)) - I_2(\mathbf{x}, \mathrm{Opt}(\mathbf{x}, \boldsymbol{\theta}_2))| \leq \delta$, con $\delta \in \mathbb{R}$ pequeño. Ver Subsección 2.2.5) y no generarán cambios en el individuo, por lo que se propone seleccionar un terminal aleatorio con estructura funcional distinta al nodo seleccionado del individuo, es decir: $\{Node(I_i, j) = B_k \mid B_k = selBF(.) \wedge Node(I_i, j) \neq B_k\}$.

  Lo anterior permite la diversificación en el espacio de búsqueda sin afectar la estructura del árbol sintáctico, cambiando únicamente la estructura funcional del modelo; sin embargo, no existe la seguridad de que el cambio sea "pequeño" debido a que cada bloque funcional corresponde a una función auto contenida que se aplica sobre todo el conjunto de rezagos $\mathbf{x}$.

- *Agregación de un árbol aleatorio*
  En esta modificación se genera un nuevo individuo I_j por medio del procedimiento del Algoritmo 2.3. El árbol I_j generado, se agrega al individuo I_i seleccionado por medio de los operadores suma {+} o multiplicación {*}, en el nodo seleccionado aleatoriamente. Siendo esta propuesta similar a la aplicación del operador genético de cruce original de GP, teniendo en cuenta que en este operador es necesario un único individuo I_i y el otro individuo I_j es generado aleatoriamente.

  Debido a que en esta propuesta de modificación del individuo I_i se adicionan múltiples bloques funcionales por medio del individuo I_j, no es posible garantizar que los cambios sean "pequeños" en el individuo; sin embargo, se espera que I_j permita la diversificación de la solución I_i en el espacio de búsqueda. Se dice que se espera, dado que la generación del árbol I_j es de manera aleatoria y podría corresponder a una combinación lineal de los subárboles ya presentes en I_i, y por ende, no agregar información adicional (cambio en la estructura funcional del individuo I_i).

- *Agregación de un árbol existente*
  La agregación de un árbol existente o parte del mismo ha sido explorada por distintos autores en el operador genético de cruce de GP [69], en el que se reemplaza el subárbol con menor valor de aptitud y se reemplaza por otro aleatorio. Por lo que, la modificación de agregación de un árbol existente, es equivalente al operador genético de cruce con modificaciones en la selección del árbol, el cual se tratará en la Subsección 2.2.8 analizando las propuestas en torno al operador genético de cruce.

▪ *Cambio en la dirección de mejora del individuo*

En la programación genética, este cambio esta dado por la aplicación del operador genético de cruce, el cual se basa en la la definición de los algoritmos genéticos (GA, por su sigla en inglés), en la que se busca realizar un cambio en las componentes del individuo en aquella dirección de mejora del valor de aptitud de los mismos.

En el algoritmo original GP no es posible garantizar que el cambio en el individuo generará mejoras en aquella dirección de mayor mejora del valor de aptitud de acuerdo con la población actual; sin embargo, si genera mejoras sucesivas basadas en la aleatoriedad (aplicación del operador genético de mutación) y en la recombinación de los individuos (operador genético de cruce).

Por lo anterior, se propone aplicar el operador genético original de mutación de GP a un individuo I_i en un nodo hoja $Node(I_i, k)$; con $k = ceil\big(random(1, size(I_i)\big)$ el número del nodo a ser seleccionado, $ceil(a)$ función que retorna un número entero correspondiente al techo de a de acuerdo con la expresión: $ceil(a) = min\{n \in \mathbb{Z} \mid n \geq a\}$, $radom(a, b)$ función que retorna un número aleatorio entre a y b, y $size(I_i)$ función que retorna el número de nodos del individuo I_i. Reemplazándolo por un bloque funcional $B_m \subseteq T$ tal que $\{\exists I_j \mid \forall_{h \neq k}\ Node\big(I_j, h\big) = Node(I_i, h) \wedge Node\big(I_j, k\big) = B_m\ \wedge Apt(I_j)\ >\ Apt(I_i)\ \wedge B_m \notin I_i\}$. En caso de que no exista un B_m que cumpla la condición, este debe ser seleccionado aleatoriamente de los bloques funcionales del mejor individuo de la población. Dicha propuesta es la más cercana a la definición del operador genético de mutación por medio de una búsqueda iterativa de bloques funcionales que permitan la mejora estructural del individuo manteniendo la estructura del árbol sintáctico equivalente a I_i, equivalente a una extensión de la definición de cambio de un terminal visto en la Subsección 2.2.7, en la que se definen restricciones al terminal seleccionado.

Considerando lo anterior y tomando en cuenta la dirección mejora de los individuos, lo que implica mejoras consecutivas a los individuos encontrados, se optó en este trabajo por esta definición del operador genético de mutación, con la salvedad de que se debe utilizar una variable aleatoria que indique si se aplica esta nueva definición o la definición tradicional, con el fin de garantizar diversificación en el espacio de búsqueda y que el algoritmo de GP no quede atrapado en zonas de múltiples mínimos locales durante el proceso de búsqueda.

2.2.8 Modificación del operador genético de cruce

El objetivo del operador genético de cruce es realizar una combinación de las características de dos individuos padre $I_i \in P_g$ y $I_j \in P_g$, con $i \neq j$, para generar otro par de individuos hijos I_i^* y I_j^* que permita la intensificación en el espacio de búsqueda por medio de la inclusión de las características de los padres originales. De acuerdo a lo anterior, el operador genético de cruce se puede expresar de la forma: $[I_i^*, I_j^*] = Cross(I_i, I_j)$, con la función Cross(.) equivalente a la aplicación del operador genético de cruce a los individuos padre I_i y I_j y cuyo resultado es un vector con los dos individuos hijos I_i^* y I_j^*.

Teniendo en cuenta la propuesta de generación de individuos de GP de la Sección 2.1, en los cuales cada nodo hoja de I_i está compuesto única y exclusivamente por bloques funcionales, considerando que los bloques funcionales son funciones auto contenidas con estructura funcional propia y evaluada sobre todos los rezagos definidos X, y que la agregación de nuevas estructuras funcionales al individuo equivale a sumar otro individuo I_j, es posible modificar al individuo I_i por medio de la agregación de otro individuo I_j en la dirección funcional de mejora del valor de aptitud entre ambos.

En la Figura 2-6 se muestra los posibles resultados del cruce entre dos individuos por medio del procedimiento tradicional del algoritmo de GP mostrado en el Algoritmo 2.7 cuando $random(0,1) \leq prbNCross$. El punto A de la Figura 2-6 corresponde a una posición arbitraria en la dirección de mejora entre los individuos, mientras que los restantes B, C y D corresponden a puntos arbitrarios posibles en la aplicación del algoritmo de cruce tradicional de GP a los individuos padre I_i y I_j.

Se propone por ende, adicionar una modificación del operador genético de cruce, en el que se genere un único individuo hijo de acuerdo a la expresión:

$$I_i^* = I_i + I_j \tag{2.15}$$

Con parámetros $\boldsymbol{\theta_i} = Opt(\{\boldsymbol{\theta_i} \cup \boldsymbol{\theta_j}\}, \dots)$. Lo anterior permite adicionar las características funcionales de mejora de cada uno de los individuos padre I_i y I_j, siendo los parámetros $\boldsymbol{\theta_i}$ los que determinarán los respectivos coeficientes (marco de referencia).

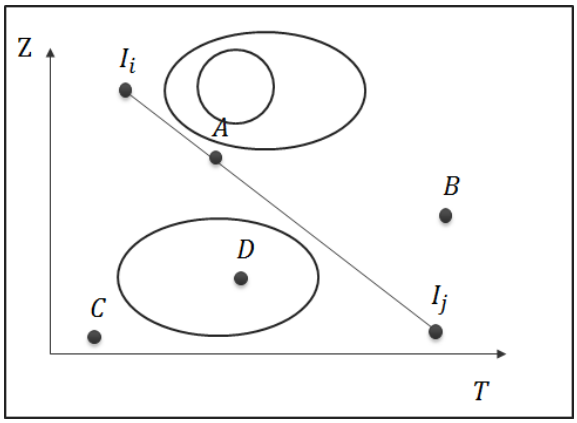


Figura 2-6: Representación de la aplicación del operador genético de cruce tradicional entre los individuos padre I_i y I_j.

En la Figura 2-7 se puede apreciar que el resultado del cruce entre los individuos padre I_i y I_j propuesto (I_i^*), el cual corresponde a un punto en la recta entre los individuos padre; que si bien no puede garantizar una dirección de mejora global, permite direccionar a los hijos a una estructura funcional mejor que los padres, debido a que $I_i \subseteq I_i^* \wedge I_j \subseteq I_i^*$.

Dado que no es posible garantizar la dirección de mejora función global de los individuos, se propone el uso de una variable aleatoria $prbNCross \in [0,1]$, que permita seleccionar cuál de las versiones del operador de cruce se aplicará, la versión tradicional o la propuesta de la ecuación 2.15.

El uso de $prbNCross$ permite la mejora sucesiva de los individuos al concentrarse en estructuras funcionales de interés —individuos padre—, y mantener la diversificación en la estructura de las soluciones (generación de individuos en nuevos puntos del espacio de búsqueda).

Algoritmo 2.7. Propuesta de operador genético de cruce.

```
01 Function Cross(I_i, I_j)
02     prbNCross
03     I_i* ← {}
04     I_j* ← {}
05     if random(0,1) ≤ prbNCross then
06         a ← ceil((random(1, size(I_i))))
07         b ← ceil((random(1, size(I_j))))
08         I_i* ← I_i
09         Node(I_i*, a) ← Node(I_j, b)
10         I_j* ← I_j
11         Node(I_j*, b) ← Node(I_i, a)
12     else
13         I_i*(X, Θ) ← (I_i + I_j)(X, Opt(I_i + I_j))
14     end if
15     return [I_i*, I_j*]
16 end function
17 end algorithm
```

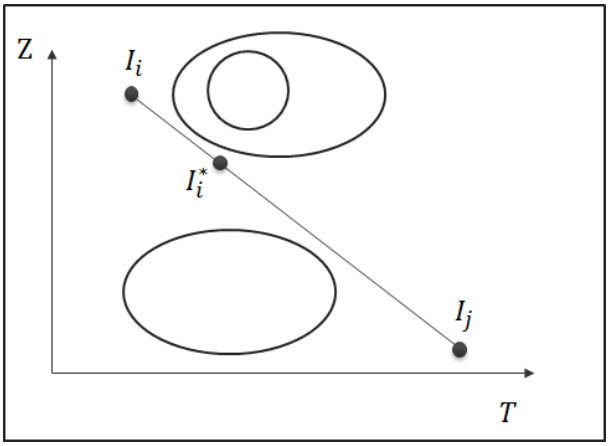


Figura 2-7: Representación de la aplicación del operador genético de cruce propuesto entre los individuos padre I_i y I_j.

En el Algoritmo 2.7 se muestra el procedimiento para la aplicación de la propuesta de cambio del operador de cruce, utilizando para ello $prbNCross$ equivalente a la probabilidad de aplicación del algoritmo tradicional de GP. De acuerdo a los siguientes pasos:

1. Inicializar una nueva variable al inicio del algoritmo de GP llamada $prbCross \rightarrow [0,1]$ que indica si se aplica o no el operador de cruce tradicional de GP.

2. Si durante el proceso de exploración del algoritmo de GP fue seleccionada la aplicación del operador genético de cruce, entonces se calcula la variable aleatoria $prbNCross \rightarrow [0,1]$:
    a. Si $prbNCross \leq \vartheta$, $\vartheta = random(0,1)$ entonces se aplica el operador tradicional de cruce.
    b. En caso contrario, se aplica el operador de adición $I_i^* = I_i + I_j$ con parámetros $\theta_i = Opt(\{\theta_i \cup \theta_j\}, \dots)$.

La propuesta anterior permite concentrar las soluciones en aquellas regiones de interés, incluyendo en el nuevo individuo aquellas componentes de mejora de sus individuos padre.

Es de tener en cuenta que la variable $prbCross$ debe ser dinámica, de acuerdo con la información del comportamiento de las poblaciones entre generación y generación, por lo que se propone su cálculo de acuerdo con: $prbCross = min(\kappa Cross_0 + \xi Cross(P_g) * \tau Cross, \kappa Cross_1)$, con $\kappa Cross_0 \in [0,1]$ una constate real que equivale a la mínima probabilidad de aplicación del operador de cruce propuesto en cualquier generación g, $\kappa Cross_1 \in [0,1]$ una constante real que corresponde a la máxima probabilidad de aplicación del operador de cruce propuesto en cualquier generación g, $\tau Cross \in \mathbb{R}$ es el porcentaje de incremento máximo de la probabilidad de aplicación del operador de cruce propuesto, $\xi Cross(P_g)$ es la función que analiza el porcentaje de mejora ponderado del total de individuos de la población por medio de la expresión: $\xi Cross(P_g) = \sum mejor(I_i, P_{g-1})/n$, con $mejor(I_i, P_{g-1}) = 1$ si $Apt(I_i) > Max\left(Apt(P_{g-1})\right)$. Esta modificación permite adaptar la focalización de los individuos en zonas de interés de acuerdo con las condiciones de los individuos de la población y características propias de la zona, ayudando a superar los problemas de falta de diversificación en regiones planas y aumento desmedido de la misma en zonas con muchos minimos locales.

2.2.9 Modificación del operador selección

El operador de selección permite seleccionar que individuos I_i conformarán la población P_{g+1} del algoritmo de GP durante el proceso de búsqueda. En la versión original del algoritmo de GP (y propuesta realizada en la tesis de maestria [12]), se reemplaza la población actual (P_g) por una nueva población generada a partir de la aplicación de los operadores genéticos definidos $P_{g+1} = P_g^*$ [3]. Lo que conlleva a perdida de información relevante de generaciones anteriores, expresadas en las formas funcionales de los individuos que lo conforman, y por ende, un aumento desmedido en la diversidicación que va en detrimento de la focalización en zonas prometedoras del espacio de búsqueda.

Debido a que la selección de los individuos determina el grado de concentración de las soluciones en regiones específicas del espacio de búsqueda, distintos autores han propuestos cambios en pro de una guía más inteligente hacia zonas prometedoras; entre las mejoras propuestas se encuentran: el uso del algoritmo de variación dinámica de la población (DPV) para la determinación del tamaño de la población dinámicamente durante el proceso de búsqueda de GP [70]; el cambio del tamaño de la población de acuerdo al valor de aptitud de los individuos que la componen [74]; la aplicación de operadores $\mu + \lambda$ para la selección de los individuos de la población de GP [53]; uso conjunto de los métodos de multipoblación y selección dinámica adaptativa para la eliminación de los individuos irrelevantes para el proceso de búsqueda de GP [77]; la determinación del tamaño del conjunto de datos de entrenamiento por medio del índice Hoeffding basado en algoritmos evolutivos (HEA) [23]; el análisis de la distancia entre funciones para la selección de los individuos basada en la probabilidad de transición multipaso (MSTP) [71]; el cambio de ubicación del paso de selección de los individuos dentro del algoritmo de GP [72]; cambio en la medida de la función de aptitud basada en la minimización del riesgo estructural (SRM) para la selección de los individuos [9]; extensión del espacio de búsqueda por medio del análisis de los vecinos de los individuos de la población de GP [73]; el mapeo

adaptativo por medio del método de Huffman [75]; y el control de dos fases para la diversidad de los individuos [76].

De los anteriores, se destaca [53], quien propone el uso de la metodología $\mu + \lambda$, en la cual, la población actual es reemplazada por los mejores n individuos de $P_g \cup P_g^*$. Lo anterior implica que se mantienen aquellos individuos con mejor medida de la función de aptitud (fitness) entre generaciones, focalizando el espacio de búsqueda en soluciones prometedoras.

Sin embargo, al analizar la propuesta de selección de individuos $\mu + \lambda$ en espacios de búsqueda semiplanos $\{\forall I_i, I_j \in A : \left|I_{\mathrm{i}}(\mathbf{x}, \mathrm{Opt}(\mathrm{X}, \boldsymbol{\theta}_i)) - I_{\mathrm{j}}(\mathbf{x}, \mathrm{Opt}(\mathrm{X}, \boldsymbol{\theta}_j))\right| \leq \delta\}$, con $\delta \in \mathbb{R}$ pequeño; implicaría que los individuos generarían soluciones muy similares (en términos del error de predicción), y por ende, la proporción de individuos de la población anterior se mantendrían durante varias generaciones, focalizando el individuo en zonas no prometedoras y restándole diversificación al proceso de búsqueda. Por lo que, en este trabajo se propone el uso de la metodología (μ, λ), en la cual, son seleccionados los mejores μ individuos de la población actual P_g y los λ mejores individuos de la nueva población generada P_g^*. Lo anterior permite diversificar la zona de búsqueda por medio de la inclusión de individuos de P_g^*, manteniendo los individuos de zonas más prometedoras de P_g.

Es de resaltar que los valores de μ y λ son asignados a priori por el investigador de acuerdo al conocimiento previo de la serie, pero en la dinámica de búsqueda del algoritmo de GP, es posible pasar de zonas planas a zonas con demasiados mínimos locales, condición que no es estática y genera un desaprovechamiento de los individuos entre generaciones. Es así, que se propone, modificar los valores de μ y λ, de acuerdo con la siguiente expresión:

$$\mu = ceil\big(min\big(\kappa_0 + \xi\big(P_g\big) * \tau, \kappa_1\big) * n\big) \tag{2.16}$$

$$\lambda = n - \mu \tag{2.17}$$

Donde $\kappa_0 \in [0, 1]$ es una constate real que equivale a la mínima proporción de selección de padres actuales en cualquier generación g, $\kappa_1 \in [0, 1]$ es una constante real que corresponde a la máxima proporción de selección de padres actuales en cualquier generación g, $\tau \in \mathbb{R}$ es el porcentaje de incremento máximo, $\xi(P_g)$ es la función que analiza el porcentaje de mejora ponderado del total de individuos de la población por medio de la expresión: $\xi\big(P_g\big) = \sum mejor(I_i, P_{g-1})\,/n$, con $mejor(I_i, P_{g-1}) = 1$ si $Apt(I_i) > Max\left(Apt\big(P_{g-1}\big)\right)$, y $ceil(a)$ es la función que retorna un número entero correspondiente al techo de a de acuerdo con la expresión: $ceil(a) = min\{n \in \mathbb{Z} \mid n \geq a\}$.

La ecuación 2.16 es muy similar a la ecuación 2.13 —de reproducción—, teniendo en cuenta que el operador de reproducción es equivalente a la selección de μ individuos de la población actual, por lo que se puede eliminar el operador genético de reproducción, por medio del operador de selección propuesto. La propuesta de cambio del operador de selección, permite la focalización de μ individuos en zonas de interés por medio del mantenimiento de estructuras funcionales especificas representadas por los individuos seleccionados en la población actual; además de la diversificación de las soluciones en el espacio de búsqueda por medio de la selección de los λ individuos de la población generada por la aplicación de los operadores genéticos de mutación y cruce.

2.2.10 Modificación del operador simplificación

El operador de simplificación permite la disminución en el tamaño de los individuos que no aportan a la predicción de la serie de tiempo (bloat) durante el proceso de búsqueda del algoritmo de GP. Varios autores han propuesto metodologías para tratar el problema del bloat que pueden ser considerados como versiones del operador de simplificación, las cuales se basan en el uso de métodos de simplificación de rango, remoción de

hijos redundantes e inclusión del operador de simplificación de regresión lineal [10]; el uso del método de Simplificación de decisión equivalente (EDS) [62]; y la simplificación algebraica a partir de reglas aritméticas [64].

Al implementar los individuos por medio de bloques funcionales, se limita automáticamente el tamaño de los individuos, reduciendo considerablemente el número de nodos, debido a que una función completa (equivalente a un individuo) se agrupa en un único nodo terminal (bloque funcional), es decir, un bloque funcional permite representar una función $G(\mathbf{x}, \mathbf{w})$ que corresponde a una componente funcional de un modelo de predicción de series de tiempo.

Analizando las propuestas mencionadas, y dado que la estructura de los individuos cambia a partir del uso exclusivo de bloques funcionales como terminales, se propone el uso de la metodología de simplificación algebraica a partir de reglas aritméticas [64], modificando las reglas para considerar la propuesta de estructura de los individuos de la Sección 2.1. A continuación se muestran los criterios de simplificación propuestos:

- *Simplificación por operador (+) y (-)*
  Si se tiene un subárbol $Node(I_i, \mathrm{k}) = B_a \pm B_b$, con $Node(B_a, 1) = Node(B_b, 1)$, y B_a y B_b polinomiales, entonces se reemplaza el subárbol por: $Node(I_i, \mathrm{k}) = B_a$, con $\boldsymbol{\theta} = Opt(.)$.

- *Simplificación de bloques funcionales autocontenidos*
  Si se tiene un subárbol $Node(I_i, \mathrm{k}) = B_a \pm B_b$, con $B_b \subseteq B_a$, entonces se reemplaza el subárbol por: $Node(I_i, \mathrm{k}) = B_a$, con $\boldsymbol{\theta} = Opt(.)$.

  Por ejemplo, para el caso particular de los bloques funcionales polinomiales, si se tiene un subárbol $Node(I_i, \mathrm{k}) = B_a \pm B_b$, con $Node(B_a, 1) = Node(B_b, 1)$, y B_a y B_b polinomiales, entonces se tiene que: $Node(I_i, \mathrm{k}) = B_a$ con $\boldsymbol{\theta} = Opt(.)$, debido a que la suma o resta de dos polinomios de igual grado es igual a otro polinomio de igual grado con parámetros distintos.

- *Simplificación de bloques funcionales tipo constante*
  Si se tiene un subárbol $Node(I_i, \mathrm{k}) = B_a \pm B_b$, con B_a un BF polinomial y B_b un BF tipo constante, entonces se reemplaza por: $Node(I_i, \mathrm{k}) = B_a$, con $\boldsymbol{\theta} = \boldsymbol{\theta}$ debido a que la constante de B_b está incluida en el bloque funcional B_a.

  Si se tiene un subárbol $Node(I_i, \mathrm{k}) = B_a\{*, /\}B_b$, con B_a un BF polinomial y B_b un BF tipo constante, entonces se tiene que: $Node(I_i, \mathrm{k}) = B_a$, con $\boldsymbol{\theta} = \boldsymbol{\theta}\,\{*, /\}\boldsymbol{\theta_b}$.

  Las anteriores reglas pueden extenderse a otras identidades matemáticas considerando el tipo de bloques funcionales a ser utilizados en los terminales.

Esta modificación propuesta al algoritmo de GP permite disminuir el bloat, aprovechando el uso de los bloques funcionales y su equivalencia con funciones de predicción de series de tiempo.

2.2.11 Modificación del operador intensificación

Se propone el uso de un operador de intensificación que permita la focalización del individuo en la estructura funcional encontrada, generando un mejor valor de aptitud por medio de la optimización en los parámetros propios del individuo a través de un algoritmo de optimización $optInt(.)$ distinto a $Opt(.)$. Por ejemplo: los algoritmos genéticos (RGNOUD en R [120]), metahurísticas, entre otros. Inicializando el punto de búsqueda con el vector $\boldsymbol{\theta}$ del individuo.

Se consideran dos alternativas para la inclusión del operador en el algoritmo de GP; la primera considera su uso al final de la ejecución del proceso de búsqueda sobre el individuo con mejor valor de aptitud en todas las generaciones; mientras que la segunda opción, considera su uso al final de cada generación sobre el mejor individuo de la misma, siendo esta última más costosa computacionalmente. Sin embargo, y considerando que

el proceso de búsqueda del algoritmo de GP intentan encontrar aquella estructura funcional (ecuación) que mejor describa los datos, que el proceso de selección de los individuos se basa en la función de aptitud de los mismos, que los parámetros de los individuos poseen gran incidencia en el valor de los mismos, y que el algoritmo de optimización $Opt(.)$ debe cumplir los criterios descritos en la Subsección 2.2.5, se optó en este trabajo por considerar incluir este operador al final del proceso de búsqueda (primera opción).

Por otro lado, se puede considerar que la función de error entre $F(\mathbf{x})$ y $\hat{F}(\mathbf{x})$ corresponde a una función nueva la cual se puede reinterpretar como una serie de tiempo $G(\mathbf{x}) = F(\mathbf{x}) - \hat{F}(\mathbf{x})$, susceptible de ser modelada de manera sucesiva aplicando el algoritmo de GP.

2.2.12 Diversificación del espacio de búsqueda

Debido a que en las secciones anteriores se ha prestado un principal interés en la intensificación y concentración de las soluciones en un espacio de búsqueda prometedor, a partir de la modificación de los operadores genéticos de cruce, mutación y reproducción (reemplazado por el operador de selección (μ, λ)); además de los operadores de selección e intensificación. En el algoritmo de GP se disminuiría en cierta medida la capacidad de exploración y diversificación en el espacio de búsqueda, aumentando la probabilidad de quedar atrapado en minimos locales.

Es así que se propone un mecanismo de diversificación basado en el reinicio de la población (generar la población de manera aleatoria) cada $rGen$ generaciones, o cada $noGen$ generaciones sin mejora del mejor individuo de la población, es decir, se aplica el reinicio de la población si: $\left\{g \equiv rGen \ \vee \ Min\left(Apt(bestInd), Apt\left(I_g^*\right), Apt\left(I_{g-1}^*\right), \dots, Apt\left(I_{g-noGen}^*\right)\right) = Apt(bestInd)\right\}$.

Lo anterior permite que el algoritmo de GP explore nuevas zonas de interés cada $rGen$ o cuando la zona de interés haya sido agotada (no se encuentre mejoras en el valor de aptitud de los individuos de la población).

2.2.13 Algoritmo propuesto

De acuerdo con las propuestas de modificación de los individuos planteadas en la Sección 2.1, y las modificaciones a los operadores de reproducción, cruce, mutación, selección, simplificación e intensificación de la subsecciones anteriores. Se propone el Algoritmo 2.8 como modificación del algoritmo original de GP para la predicción de series de tiempo, en el que se limita el espacio de búsqueda, disminuyendo el bloat y aumentando las capacidades de predicción del mismo, por medio de la recopilación de los cambios propuestos en esta sección.

La generación de la población inicial P_0 se realiza a partir del Algoritmo 2.5. La función $Cross(.)$ equivale a la definición del Algoritmo 2.7. La función $Mutate(.)$ corresponde a la aplicación del operador genético de mutación propuesta en la Sección 2.2.5. $prbCross$ es una constante real entre 0 y 1 que indica la probabilidad de aplicación del operador genético de mutación a un individuo especifico. La función $stopCriteria(.)$ evalúa los criterios de parada y retorna $true$ cuando se cumple alguno y $false$ en caso contrario; algunos de los criterios de parada más utilizados corresponden a: $g > G$, con G equivalente al número máximo de generaciones; $\left|Apt\left(bInd_g\right) - Apt\left(bInd_{g-1}\right)\right| < \delta$, con δ pequeño y $g > 1$, suele utilizarse para asegurar una mejora mínima de δ entre generaciones, algunas modificaciones incluyen el conteo de veces en el que no mejora durante un numero de generaciones específicas gM, equivalente a:

$$\sum_{k=g}^{g-gM} \begin{Bmatrix} 1, si \left|Apt\left(bInd_g\right) - Apt\left(bInd_{g-1}\right)\right| < \delta \\ 0, en\ caso\ contrario \end{Bmatrix} \geq mM \tag{2.18}$$

Donde mM es el número máximo de generaciones en los cuales no existe una mejora de al menos δ entre los mejores individuos de cada generación sucesiva en un número de gM generaciones. La función $Sel(.)$ equivale al operador de selección propuesto (μ, λ) de la Subsección 2.2.7.

Algoritmo 2.8. Algoritmo propuesto de GP.

01 $inicializar\ T, \mathrm{Z}, \boldsymbol{\theta}, prbCross, \mu, \lambda$
02 $generar\ la\ población\ inicial\ P_{g=0}$
03 $bInd \leftarrow \{\}$
04 $g \leftarrow 1$
05 **loop while** $stopCriteria(\mathbf{x}, bInd, g) = false$
06 $a \leftarrow random(0,1)$
07 $P_g^* \leftarrow P_g^* \cup Cross(I_i, I_{i+1}), \{\forall_i \mid random_i(0,1) \leq prbCross\}$
08 $P_g^* \leftarrow P_g^* \cup Mutate(I_i), \{\forall_i \mid random_i(0,1) > prbCross\}$
09 $P_g^* \leftarrow Opt(P_g)$
10 $P_g \leftarrow Sel(\mu, \lambda, P_g, P_g^*)$
11 $bInd \leftarrow bInd \cup optInt(I_k^*)$ tal que $\forall I_i \in P_g, Apt(I_k) \geq Apt(I_i)$
12 $g \leftarrow g + 1$
13 **end loop**
14 **end algorithm**

2.2.14 Cambios en la implementación

Además de los cambios conceptuales propuestos en las subsecciones anteriores, también se proponen dos cambios en la implementación tradicional de GP (y principales modificaciones) en la evaluación del valor de aptitud de cada uno de los n individuos de la población a partir de un vector de rezagos $\mathbf{x} = \{x_{t-1}, \dots, x_{t-p}\}$, debido a que este proceso consituye uno (si no es el más) costoso de los pasos del algoritmo de GP, dado que es necesario recorrer cada nodo de cada árbol sintáctico ($I_{1 \leq i \leq n}$) realizando las respectivas operaciones en postorden. Y si consideramos además, que existe un vector $\mathbf{x}^{(j)}$ para cada una de las observaciones $j = \{1, \dots, nObs\}$, los anteriores pasos se deberán por ende ejecutar $nObs$ veces. Los cambios propuestos, cuyo desarrollo en R están consignados en el Anexo C, son:

- *Cambio en el vector x de evaluación de los individuos*
  Se propone la evaluación matricial de los árboles sintácticos, por lo que $\mathbf{x} = \{x_{t-1}, \dots, x_{t-p}\}$ vector de rezagos, se convierta en $\mathbf{X} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(nObs)}\}$ con $\mathbf{x}^{(j)} = \{x_{t-1}^{(j)}, \dots, x_{t-p}^{(j)}\}$, por lo que el resultado de la evaluación de un individuo corresponda a un vector de resultados de valor de aptitud: $Apt(I_i) = \{I_i(\mathbf{x}^{(1)}, \boldsymbol{\theta}), \dots, I_i(\mathbf{x}^{(nObs)}, \boldsymbol{\theta})\}$. El cual reduce el número de recorridos de los nodos de los individuos en al menos: $nObs * \sum_{i=1}^{n} size(I_i)$.

- *Cambio en el recorrido del árbol sintáctico*
  Debido a que los lenguajes de programación actuales permiten la evaluación de expresiones regulares (armadas adecuadamente para el compilador especifico), es posible por ende asocuar una función $funEval$ a cada individuo I_i tal que corresponda con el resultado de recorrer todos los nodos del mismo. Es así que, ya no es necesario recorrer los nodos, sino tan solo realizar un llamado a una función. Por ejemplo: Consideremos el individuo $I = B_1 * B_4 + B_1$ equivaldría al llamado $sum(mul(B\_1(\dots,\mathbf{X}), B\_4(\dots,\mathbf{X})), B\_1(\dots,\mathbf{X}))$, con $sum(a, b)$ función que retorna el resultado de $a + b$; $mul(a, b)$ función que retorna el resultado de $a * b$; B_1 y B_4, funciones que retornan el resultado del calculo de los bloques funcionales B_1 y B_4 respectivamente.

Los anteriores cambios propuestos permiten un menor tiempo de ejecución del algoritmo, además de un uso más eficiente de la memoria y recursos computacionales asociados. Debido principalmente a la reducción de pasos y a la anidación de bloques de memoria temporales (en los llamados de función) sin condiciones adicionales.

2.3 Ejemplos de representación de individuos de series de tiempo con función de generación conocida utilizando la metodología propuesta

En esta sección se muestran algunos ejemplos de representación de series de tiempo con ecuación de generación conocida por medio del uso del algoritmo propuesto en la Sección 2.2 (Algoritmo 2.8), con el fin de analizar las capacidades de la metodología propuesta en el retorno de la ecuación de generación (estructura).

La implementación del Algoritmo 2.8 fue en R, cuyas funciones y procedimientos pueden ser revisados en la sección de Anexos.

Para cada serie de tiempo analizada, se ejecutó el Algoritmo 2.8 con los siguientes parámetros: $n = 20$ individuos, $\varkappa_{max} = 3$, $H_{max} = 7$ nodos, $G = 10$, $Apt(.) = SSE$, $Opt(.) = OPTIM$, $optInt(.) = RGNOUD$, $prbCross = 0.5$, $\mu = 0.1 * n$, $\lambda = 0.9 * n$, $T = \{B_1, \dots, B_{30}\}$, Z = $\{+, -, *, /\}$, $d = 1$. Los datos de p y q son definidos en cada una de las series.

2.3.1 Serie de tiempo con un único rezago

El modelo autoregresivo (AR) constituye uno de más utilizados en la predicción de series de tiempo en la literatura en general, este modelo es de la forma [13]:

$$x_t = c + \sum_{i=1}^{p} \varphi_i x_{t-i} + \varepsilon_t \tag{2.19}$$

La cual puede ser expresada de acuerdo con la propuesta de bloques funcionales como:

$$x_t = B_1 + B_3 = B_{25} \tag{2.20}$$

Para el caso particular de $p = 1$ y $c = 1$:

$$x_t = x_{t-1} + 1 = B_{25} \tag{2.21}$$

De acuerdo con la metodología propuesta en las secciones 2.1 y 2.2, es posible deducir que el árbol que describe este modelo consta de un solo nodo de tipo terminal (B_{25}), menor a los 3 nodos de su representación tradicional para un $p = 1$. En general, para modelos AR, la metodología propuesta corresponderá a un único nodo terminal; en cambio, para el algoritmo tradicional de GP corresponderá a $2^p + 1$ nodos, donde p es el número de rezagos a ser considerados en el modelo.

Realizando la verificación de la ejecución del algoritmo original de GP, en los que los terminales de los individuos corresponden a los rezagos considerados, se obtiene una solución de la forma:

$$I^*_{GP} = C_1 + C_2 + C_3 + C_4 - C_5 * X_{t-1} \tag{2.22}$$

La cual consta de 11 nodos, de los cuales cinco corresponden a coeficientes C_i, un nodo para el rezago X_{t-1} y los restantes cinco a operadores aritméticos básicos $\{+, -, *\}$, en cuyo caso se toma en cuenta el último rezago con una cantidad redundante de operadores y terminales.

Por otra parte, con la metodología planteada, ejecutando el Algoritmo 2.8, se genera en la primera iteración funciones a dos niveles (3 nodos como máximo) entre las cuales se encuentra el bloque funcional B_{25} que corresponde con el mejor modelo:

$$I^*_{propuesta} = B_{25} \tag{2.23}$$

Con $p = 1$ y $\boldsymbol{\theta} = \boldsymbol{\varphi} = [1\ 1]$, el cual registra el menor error de aproximación ($SSE = 0$) durante las iteraciones del algoritmo (desde $g = 1$) y corresponde a un único nodo (bloque funcional AR) que describe totalmente el modelo de generación de los datos.

2.3.2 Serie de tiempo a partir de un modelo autorregresivo

En este ejemplo, se utiliza un modelo AR equivalente a la ecuación 2.19, el cual puede ser expresado como la ecuación 2.20, que para el caso particular de $p = 2$ y $c = 0$ se tiene:

$$x_t = x_{t-1} - 0.91 * x_{t-2} = B_1 = B_{25} \tag{2.24}$$

El cual corresponde a una serie de tiempo con periodo constante y amplitud decreciente proporcionalmente en el tiempo [13].

De acuerdo con la ecuación 2.24, la mejor solución generada por la aplicación del Algoritmo 2.8 en el cual se utilizan bloques funcionales, corresponde a B_{25}, con periodo constante y la respectiva amplitud que desciende hasta un límite cercano a 0.

Realizando la verificación de la ejecución del algoritmo original de GP con 110 datos (los primeros 100 de entrenamiento y los restantes 10 de validación) se generan individuos en los que sus nodos terminales corresponden a los respectivos rezagos, el cual genera el mejor individuo:

$$I^*_{GP} = (C_1 * x_{t-2} * C_2 * x_{t-1} * C_3 * x_{t-1} + C_4) * C_5 * x_{t-2} \tag{2.25}$$

El cual consta de 17 nodos, de los que cinco corresponde a coeficientes, dos a rezagos y los restantes ocho a operadores aritméticos básicos $\{+,\ *\}$, en cuyo caso se toma en cuenta los dos últimos rezagos en una combinación no lineal, que si bien posee un SSE bajo, la estructura de la solución encontrada por el algoritmo tradicional de GP no es cercana al modelo de generación de los datos (estructura funcional)

Por otra parte, con la metodología planteada, ejecutando el Algoritmo 2.8, se genera en la primera iteración funciones a dos niveles entre las cuales se encuentra el bloque funcional B_1 que correspondió con el mejor modelo:

$$I^*_{propuesta} = B_1 \tag{2.26}$$

El cual registra el menor error de aproximación durante las iteraciones del algoritmo (desde $g = 1$) y corresponde a un modelo con un único nodo (bloque funcional AR) con $p = 2$ y $\Theta = \varphi = [1\ -0.91\ 0]$ que describe totalmente el modelo de generación de los datos con $SSE = 0$ y forma funcional equivalente.

2.3.3 Serie de tiempo a partir de un modelo STAR(p)

Los modelos STAR constituyen un gran avance en el manejo de funciones determinadas por umbrales, en las cuales la función de transición $F(.)$ soluciona los problemas de los modelos puramente condicionales como los SETAR ponderando los modelos AR, por medio de la expresión:

$$x_t = c_0 + \sum_{i=1}^{p} \varphi_i x_{t-i} + F\left(\frac{x_{t-d} - \Delta}{s}\right)\left(c_1 + \sum_{i=1}^{p} \theta_i x_{t-i}\right) + \varepsilon_t \tag{2.27}$$

El cual puede ser expresado por medio de bloques funcionales como:

$$x_t = B_7 + B_1 + B_8 * (B_7 + B_1) + B_3 \tag{2.28}$$

donde $F(.)$ es la función de transición (puede ser logística, exponencial, distribución acumulativa, entre otras), d es el rezago máximo a considerarse, s es la escala del modelo, Δ es la posición del modelo de transición [104-106].

Para el caso particular en la que la función de transición es igual a un modelo AR se tiene:

$$x_t = c_0 + \sum_{i=1}^{p} \varphi_i x_{t-i} + \sum_{i=1}^{p} \emptyset_i x_{t-i}\left(c_1 + \sum_{i=1}^{p} \theta_i x_{t-i}\right) + \varepsilon_t \tag{2.29}$$

Con parámetros: $c_0 = 1, p = 2, \varphi = [0.2, 1.5], \emptyset = [0.5, 0.6], c_1 = 1, \theta = [0.1,0.3], \varepsilon_t = 0$, la ecuación 2.29 es equivalente a:

$$\begin{aligned} x_t = 1 + 0.2 * x_{t-1} + 1.5 * x_{t-2} + (0.5 * x_{t-1} + 0.6 * x_{t-2}) \\ * (1 + 0.1 * x_{t-1} + 0.3 \ * x_{t-2}) \end{aligned} \tag{2.30}$$

Analizando los resultados de la ejecución del algoritmo original de GP, este genera individuos en los cuales sus nodos terminales corresponden a los respectivos rezagos, pero en una estructura funcional que no corresponde con el modelo STAR planteado:

$$I^*_{GP} = C_1 * x_{t-1} - C_2 * x_{t-2} + C_3 \tag{2.31}$$

La cual consta de nueve nodos, de los cuales tres corresponde a coeficientes, dos a rezagos y los restantes cuatro a operadores aritméticos básicos $\{+, -, *\}$, en cuyo caso se toma en cuenta los dos últimos rezagos en una combinación lineal que si bien posee un comportamiento similar ($SSE = 0.01$) no corresponde al modelo de generación de los datos (en estructura y comportamiento general).

Por otra parte, con la metodología planteada, ejecutando el Algoritmo 2.8, se genera soluciones de la forma:

$$I^*_{propuesta} = B_{23} - (B_{23} + B_{25}) - \{B_2 - (B_2 + B_2)\} * B_2 \tag{2.32}$$

El cual es equivalente a ecuación 2.33:

$$\begin{aligned} x_t = \Bigg(x_{t-1} - \left(x_{t-1} + c_0 + \sum_{i=1}^{p} \varphi_i x_{t-i} \right) \Bigg) \\ - \Bigg\{ \Bigg[\sum_{i=1}^{p} \theta_{i,1}(x_{t-i} - x_{t-i-1}) \\ - \left(\sum_{i=1}^{p} \theta_{i,2}(x_{t-i} - x_{t-i-1}) + \sum_{i=1}^{p} \theta_{i,3}(x_{t-i} - x_{t-i-1}) \right) \Bigg] \\ * \sum_{i=1}^{p} \theta_{i,4}(x_{t-i} - x_{t-i-1}) \Bigg\} \end{aligned} \tag{2.33}$$

Con $\boldsymbol{\theta} = [\sigma, \emptyset, c, \theta] = [-0.07, 0.759, -1.155, 0.228, 0.325, 0.588, 0.329, 1.087, 0.1441,\ 0.0179, 0.301]$. El cual al ser simplificado genera:

$$x_t = \left(x_{t-1} - x_{t-1} - c - \sum_{i=1}^{p} \varphi_i x_{t-i} \right) - \left\{ \left[\sum_{i=1}^{p} \theta_{i,1}(x_{t-i} - x_{t-i-1}) - \sum_{i=1}^{p} \theta_{i,2}(x_{t-i} - x_{t-i-1}) - \sum_{i=1}^{p} \theta_{i,3}(x_{t-i} - x_{t-i-1}) \right] * \sum_{i=1}^{p} \theta_{i,4}(x_{t-i} - x_{t-i-1}) \right\} \tag{2.34}$$

La ecuación 2.34 es equivalente a:

$$x_t = \left(-c - \sum_{i=1}^{p} \varphi_i x_{t-i} \right) - \left\{ \left[\sum_{i=1}^{p} \theta_{i,1} x_{t-i} - \sum_{i=1}^{p} \theta_{i,1} x_{t-i-1} - \sum_{i=1}^{p} \theta_{i,2} x_{t-i} + \sum_{i=1}^{p} \theta_{i,2} x_{t-i-1} - \sum_{i=1}^{p} \theta_{i,3} x_{t-i} + \sum_{i=1}^{p} \theta_{i,3} x_{t-i-1} \right] * \sum_{i=1}^{p} \theta_{i,4}(x_{t-i} - x_{t-i-1}) \right\} \tag{2.35}$$

No es posible la cancelación directa de las sumatorias dado que los coeficientes de cada rezago son distintos, pero la suma o resta de sumatorias generan otra de la misma estructura pero distintos coeficientes, por lo que:

$$x_t = c_0 + \sum_{i=1}^{p} \varphi_i x_{t-i} + \left\{ \left[\sum_{i=1}^{p} \theta_{i,5} x_{t-i} + \sum_{i=1}^{p} \theta_{i,6} x_{t-i-1} \right] * \sum_{i=1}^{p} \theta_{i,4}(x_{t-i} - x_{t-i-1}) \right\} \tag{2.36}$$

Equivalente a:

$$x_t = c + \sum_{i=1}^{p} \varphi_i x_{t-i} + \left\{ \left[\sum_{i=1}^{p} \theta_{i,5} x_{t-i} + \sum_{i=1}^{p} \theta_{i,6} x_{t-i-1} \right] * \left[\sum_{i=1}^{p} \theta_{i,4} x_{t-i} - \sum_{i=1}^{p} \theta_{i,4} x_{t-i-1} \right] \right\} \tag{2.37}$$

Equivalente a la expresión:

$$x_t = c_0 + \sum_{i=1}^{p} \varphi_i x_{t-i} + \sum_{i=1}^{p+1} \theta_{i,7} x_{t-i} * \left(\sum_{i=1}^{p+1} \theta_{i,8} x_{t-i} \right) \tag{2.38}$$

El cual corresponde con la estructura del modelo generador $STAR(AR(p = 2))$, cabe resaltar que la notación de parámetros $\theta_{\mathrm{i,j}}$ equivale a la variable $\theta_{*,\mathrm{j}}$ para el rezago i. Que en forma de bloques funcionales es equivalente a:

$$I^*_{propuesta} = B_{25} + (B_1 * B_1) \tag{2.39}$$

2.4 Conclusiones

En este capítulo se mostró conceptualmente que es posible la inclusión de conocimiento experto presente en los modelos de predicción de series de tiempo de la literatura actual por medio del uso de bloques funcionales, logrando restringir el espacio de búsqueda al aumentar la probabilidad de selección de los distintos bloques funcionales a partir de las propiedades estadísticas de la serie de tiempo.

Se mostró que es posible la eliminación de los terminales tipo constante (dinámicas y estáticas) por medio del uso de bloques funcionales, permitiendo la focalidación del espacio de búsqueda hacia las componentes estructurales de los individuos, sin limitarlos por los parámetros/constantes de optimización; implementando una forma de inclusión de todos los rezagos en todos los individuos de la población de GP, permitiendo que la exploración en el campo de búsqueda correspondiera a mejoras estructurales y no de la incorporación de rezagos particulares.

El uso de bloques funcionales en el algoritmo de GP provee mejoras estructurales en la clase de individuos que genera (representación), manejo de rezagos e interrelación de modelos (cada terminal corresponde a una función de un modelo de predicción de series de tiempo de la literatura), lo que permite garantizar individuos lógicos de manera matemática (por medio de la agrupación de bloques funcionales con características matemáticas similares), y su posterior interpretación a la luz de las variables propias de modelación.

La predicción de series de tiempo a partir de la metodología propuesta permite la mezcla de modelos tradicionales de una manera sucesiva entre iteraciones, logrando así la incorporación de las particularidades propias de cada uno de ellos en un nuevo modelo con características de aproximación mayor a la de los originales (en términos de error, tendencia y estructura) con una representación matemática más sencilla.

Se modificaron los operadores genéticos de cruce y mutación, permitiendo un desarrollo más acorde con su definición original, permitiendo una mezcla entre la diversificación del espacio de búsqueda (explosión) y la focalización en zonas de interés.

Se introdujo el operador de simplificación que permite la eliminación de información innecesaria en los modelos, por medio de la poda de los árboles de acuerdo con operaciones aritméticas.

Por otra parte, también fue introduccida una función de aptitud basada no solo en criterios de error de predicción, sino, incorporando información relacionada con el tamaño de los individuos, permitiendo seleccionar aquellos que realmente aporten a la predicción de la serie de tiempo con el menor costo computacional posible.

Lo anterior complementándose con el operador de selección, el cual permite mantener memoria de aquellas características estructurales de los mejores individuos (de acuerdo con su valor de aptitud) entre generaciones, y diversificándolas por medio de las nuevos individuos generados a partir de los operadores de cruce y mutación empleados; logrando así, la eliminación del operador de reproducción del algoritmo original de GP.

En las series de datos analizadas en la Sección 2.3, es posible observar que el modelo modificado de GP logra hallar el modelo generador de los datos de una manera más directa, con una estructura similar y con un menor número de iteraciones y nodos redundantes, mejorando los tiempos de ejecución, la jerarquía de las soluciones encontradas y disminuyendo el error de aproximación.

3. Aplicación de la metodología propuesta a la predicción de series de tiempo conocidas

3.1 Introducción

La necesidad de contar con herramientas orientadas a la toma adecuada de decisiones, ha conducido en las últimas décadas a un acelerado interés por el uso de técnicas de inteligencia computacional en el desarrollo de modelos de pronóstico de series de tiempo [1], debido principalmente a su flexibilidad para aprender relaciones no lineales desconocidas; estas técnicas incluyen las redes neuronales artificiales [2], los sistemas de inferencia borrosa [2], los sistemas neurodifusos, y las técnicas evolutivas como la programación genética [3].

Entre los principales modelos estadísticos se resalta el modelo ARIMA desarrollado por Box & Jenkins [13], en el cual se asume linealidad entre las variables (rezagos) utilizadas; sin embargo, lo anterior constituye una falencia para incorporar relaciones no lineales entre las variables. Para solventar dicho problema, se han desarrollado modelos no paramétricos y de inteligencia computacional como las redes neuronales artificiales (ANN, por su sigla en inglés), las cuales se basan en el uso de neuronas para aproximar las funciones [2]; los perceptrones multicapa (MLP, por su sigla en inglés), el cual es una de las implementaciones más comunes de ANN compuesta por un numero finito de capas en las que cada neurona está relacionada con una única neurona en la capa anterior y siguiente a través de las sinapsis definidas [2]; el modelo de Arquitectura dinámica para redes neuronales (DAN2, por su sigla en inglés) que corresponde a una implementación de las ANN en el que el conocimiento se transfiere de capa a capa de la ANN por medio de nodos CAKE [16]; las máquinas de vector de soporte (SVM, por su sigla en inglés) las cuales por medio de un kernel definido permiten clasificar los datos y predecir a que grupo pertenece las nuevas muestras [17]; y el algoritmo original de programación genética (GP, por su sigla en inglés) en el que por medio de individuos (que representan ecuaciones matemáticas) y la aplicación de operadores genéticos es posible la generación de modelos de predicción [3].

El algoritmo de programación genética puede entenderse como una extensión de los algoritmos genéticos en el que los individuos representan expresiones matemáticas [3]. Para ello, dichas expresiones matemáticas son representadas como árboles sintácticos donde las funciones y operadores matemáticos (suma, resta, multiplicación, etc.) van en los nodos interiores (o nodos funcionales), y las variables y constantes numéricas van en los nodos terminales. De esta forma, al aplicar los operadores genéticos de cruce y mutación sobre una población de individuos, se realiza una búsqueda en el espacio de expresiones matemáticas factibles con el fin de encontrar una ecuación empírica que permita representar la relación existente entre un conjunto de variables de entrada y una variable de salida; en este caso, la GP recibe el nombre de regresión simbólica (SR, por su sigla en inglés) [3].

En su versión original, el algoritmo de GP propuesto por Koza [3], corresponde a:

1. En la iteración o generación inicial ($g = 0$) se crea una población (P_0) de n individuos.
2. Para cada individuo S_i, con $i = \{1, \ldots, n\}$, se evalúa la función de aptitud $f(S_i)$.
3. Se genera una nueva población P_g^* aplicando los operadores genéticos a la población actual P_g, de la siguiente forma:

a. Son seleccionados los padres a los cuales se le aplicará los operadores de cruce y clonación de una manera probabilística.
b. A diferencia de los algoritmos genéticos tradicionales no se considera el operador de clonación por lo que se limita a cruce y mutación.
c. La población de hijos P_g^* reemplaza la población actual P_g así: $P_g = P_g^*$.

4. Se evalúan los criterios de parada (usualmente es utilizado el número máximo de generaciones), sino se cumplen se vuelve al paso 2. En caso contrario se termina la ejecución del algoritmo.

En esta tesis se propone un nuevo algoritmo de predicción de series de tiempo basado en el algoritmo original de GP, en el cual son cambiados los terminales a ser considerados (constituidos única y exclusivamente por bloques funcionales), se incluyen nuevos pasos al algoritmo (inclusión de los operadores de simplificación e intensificación) y se redefinen los operadores de reproducción, cruce, mutación, y selección (por medio del uso del criterio (μ, λ)).

Los anteriores cambios pueden apreciarse en el Algoritmo 2.8 de la Subsección 2.2.13, donde se resalta el paso previo de análisis estadístico de la serie de tiempo para la inicialización de los parámetros a ser utilizados, la eliminación del operador genético de reproducción por medio del uso del operador de selección (μ, λ), la aplicación de la función $optInt(.)$ equivalente al operador de intensificación, y el cambio en la definición de los operadores genéticos de mutación y cruce (modelo de agregación de individuos) para la focalización de los individuos en regiones de interés.

Los cambios propuestos están fundamentados en el cambio del procedimiento de generación de los individuos a partir de la inclusión de bloques funcionales; la modificación de los operadores de selección para la concentración de la población en zonas de interés; el uso del operador de intensificación para la optimización de los parámetros del individuo, sin modificar su estructura funcional; la aplicación del operador de simplificación y el cambio de la función de aptitud para la reducción del bloat durante el proceso de búsqueda; modificación de la probabilidad de aplicación del operador genético de reproducción proporcional al comportamiento de la población de individuos durante las últimas generaciones; modificación del operador de cruce para la concentración de soluciones en zonas de interés incorporando la información de ambos padres en un único individuo hijo; la modificación al operador de mutación que permita introducir pequeños cambios en la estructura del individuo en dirección de una mejora en el valor de aptitud del mismo; y la diversificación del espacio de búsqueda a partir del reinicio adaptativo de la población de individuos durante el proceso de búsqueda.

Con el objetivo de focalizar las soluciones en zonas de interés en el espacio de búsqueda en equilibrio con la diversificación del mismo, la diminución del bloat y una búsqueda más dirigida, garantizando el uso de todos los rezagos de interés, y la incorporación de los principales modelos de predicción de series de tiempo de la literatura en las posibles soluciones encontradas. Sin embargo, es necesaria la validación de las mejoras al algoritmo original propuestas en este trabajo, para lo cual, se propone la comparación de los resultados obtenidos contra series benchmark de predicción de series de tiempo analizadas con otras técnicas. Por lo que, el objetivo de este capítulo es el logro del objetivo específico:

> *Implementar el algoritmo propuesto de programación genética, y validar sus bondades por medio de simulación contra series benchmark de la literatura de predicción de series de tiempo, comparando el error de predicción, la inclusión de los rezagos definidos, y la identificación de las componentes de los modelos de predicción de series de tiempo.*

Para cumplir con el objetivo del capítulo, fueron seleccionadas cinco series benchmark que han sido pronosticadas por distintos autores utilizando modelos ARIMA, GP, MLP, ANN, DAN e híbridos. El proceso de prueba y los resultados obtenidos son discutidos en la Sección 3.2, donde se incluye la cantidad de datos empleados para el entrenamiento y la predicción, rango de fechas, transformaciones empleadas y los

respectivos resultados obtenidos. En la Sección 3.3 son presentadas las conclusiones de este capítulo. Y en la Sección 3.4 corresponde a las conclusiones.

3.2 Metodología empleada

En el Capítulo 2, fueron presentadas una serie de mejoras al algoritmo original de GP, las cuales se basan en el análisis de las propiedades estadísticas de la serie de tiempo y su incorporación por medio de un vector de parámetros generales. Por lo que, en esta sección son analizados los parámetros utilizados durante el proceso de búsqueda del algoritmo, las distintas funciones de optimización y criterios de parada utilizados en la predicción de las series benchmark seleccionadas.

3.2.1 Criterios de parada

En la Subsección 2.2.10 se definieron los criterios de parada:

- *Máximo número de generaciones* G, en el que si $g \geq G$ entonces $stopCriteria(.) = true$.
- *Mínima mejora entre generaciones* δ, en el que si $\left|Apt\left(bInd_g\right) - Apt\left(bInd_{g-1}\right)\right| < \delta$ entonces $stopCriteria(.) = true$. Con δ pequeño, $g > 1$, $Apt(.)$ corresponde a la función de aptitud, y $bInd$ corresponde al vector de mejores individuos de cada generación.
- *Número máximo de generaciones consecutivas* nM sin cumplir el criterio de mínima mejora entre generaciones $\boldsymbol{\delta}$, en el que si $\sum_{k=g}^{g-gM} \begin{Bmatrix} 1, si \left|Apt\left(bInd_g\right) - Apt\left(bInd_{g-1}\right)\right| < \delta \\ 0, en\ caso\ contrario \end{Bmatrix} \geq mM$, entonces $stopCriteria(.) = true$. Con mM equivalente al número máximo de generaciones en los cuales no existe una mejora de al menos δ entre los mejores individuos de cada generación sucesiva en un número de gM generaciones. La función $Sel(.)$ equivale al operador de selección propuesto (μ, λ) de la Subsección 2.2.9.

Para las series analizadas, se utilizó el criterio de parada estándar de máximo número de generaciones $G = 100$, debido principalmente a que los operadores genéticos de cruce y mutación generan nuevos individuos por medio de la agregación (suma) de los individuos padre, y que cada individuo está constituido por un bloque funcional, por lo que el algoritmo tiende a converger a estructuras funcionales especificas al cabo de pocas generaciones. Adicionalmente, $G = 100$ corresponde a un número estándar de generaciones en la aplicación del algoritmo original de GP [3] y constituye una base comparable en la eficiencia del mismo.

3.2.2 Algoritmos de optimización empleados

Dadas las características que debe cumplir el algoritmo de optimización $Opt(.)$ descritas en la Subsección 2.2.5, se optó por utilizar el algoritmo OPTIM [109] para la validación contra las series benchmark seleccionadas, el cual realiza una optimización basada en gradiente, que garantiza que dado un punto inicial a, para un individuo I_i, siempre convergerá a un valor similar de aptitud $Apt(I_i)$, esto es equivalente a: dados $a = b$ entonces $\left|Apt\left(I_i(\mathbf{x}, Opt(a, \ldots))\right) - Apt\left(I_i(\mathbf{x}, Opt(b, \ldots))\right)\right| \leq \delta$, con δ pequeño.

En este capítulo se utilizó el método por defecto "SANN" del algoritmo OPTIM, el cual corresponde a la técnica de recocido simulado (SA, por su sigla en inglés) propuesto por Belisle [109]. En la implementación realizada en el algoritmo OPTIM, se utiliza la función Metropolis para definir la probabilidad de aceptación de cada punto. Por defecto, los puntos candidatos son generados por medio de un Kernel Gausiano de Markov con escala proporcional a la temperatura, la cual decrece por medio de un patrón logarítmico [109].

Para el operador de intensificación mostrado en la Subsección 2.2.11, se utilizó el algoritmo RGNOUD [120], el cual combina algoritmos evolutivos con los métodos de gradiente quasi-Newton para encontrar los

parámetros de optimización que generan el mejor valor de aptitud, para lo cual se utiliza como punto de búsqueda inicial los parámetros $\boldsymbol{\theta}$ encontrados previamente durante el proceso de búsqueda del individuo I_i.

3.2.3 Probabilidad de aplicación de los operadores genéticos

En el Capítulo 2, son presentadas una serie de modificaciones a los pasos algoritmo original de GP; además de la probabilidad de selección de los operadores genéticos, las cuales pasan de ser estáticas a dinámicas, ajustándose a las características de la población actual representada en la mejora en la aptitud de sus individuos.

De acuerdo a la propuesta presentada en la Subsección 2.2.7, en la que se reemplaza el operador de selección $P_{g+1} = P_g^*$ [3] por $P_{g+1} = P_g^{\mu} \cup P_g^{*\lambda}$, donde μ es calculado en cada generación por medio de la ecuación 2.16 y λ corresponde a su complemento (ecuación 2.17). Para realizar el cálculo iterativo de μ, es necesario definir los valores de κ_0, κ_1 y τ, los cuales para la validación de series de tiempo analizadas fueron seleccionados: $\kappa_0 = 0.1$ (equivalente al 10% de la población, la cual corresponde a la probabilidad de aplicación del operador genético de reproducción de la versión original del algoritmo de GP sugerido por Koza [3]), $\kappa_1 = 0.5$ (máximo el 50% de la población para garantizar un equilibrio entre la diversificación representada por la nueva población generada y la focalización representada por la población ya existente) y $\tau = (\kappa_1 - \kappa_0)/G$. Este operador de selección reemplaza al operador genético de reproducción, en la cual ya no existe una probabilidad de aplicación del operador de reproducción, sino la certeza de la selección de al menos un porcentaje κ_0 de individuos.

Debido a que la probabilidad de aplicación del operador genético de reproducción es igual a 0, se propone para este trabajo el uso de una probabilidad de aplicación de los operadores genéticos de cruce y mutación equivalente a 0.5, es así que $prbCross = 0.5$, lo que permite un equilibrio entre la aplicación del operador genético de cruce y el de mutación.

Adicionalmente, dado que en la Subsección 2.2.8 se propone el uso de una variable aleatoria para la selección del operador genético de cruce a ser utilizado (versión original o modelo de agregación propuesto), el cual depende del valor de la probabilidad de selección $prbNCross$ definida, se propone para el análisis de las series presentadas en este capítulo el uso de $prbNCross = 0.5$, el cual corresponde a una versión del algoritmo equilibrada entre el cruce aleatorio y la focalización de los individuos en aquella línea estructural entre los individuos padre.

3.2.4 Función de aptitud empleada

De acuerdo con la función de aptitud propuesta en la Subsección 2.2.2, se utilizará el criterio de información Akaike modificado para la versión del algoritmo propuesto de GP (con uso de bloques funcionales), presentado en la ecuación 2.12. Aunque para la comparación con los demás modelos de la literatura de predicción de series de tiempo se calcularan los siguientes indicadores:

- *Suma del error cuadrático (SSE, por su sigla en inglés)*
  Este indicador realiza una sumatoria de las diferencias cuadráticas entre los resultados generados por el individuo y los valores reales registrados por la serie de tiempo. Considerando la propuesta realizada en el Capítulo 2, es equivalente a la siguiente expresión:

$$Apt(I_i) = SSE(I_i) = \sum (I_i(\mathbf{x}, \boldsymbol{\theta}) - F(\mathbf{x}, \boldsymbol{\theta}))^2 \tag{3.1}$$

  Este indicador suele ser utilizado en la comparación de los resultados entre modelos de predicción de series de tiempo por su facilidad de cálculo y resultado neto de error. Aunque se debe considerar que no toma en consideración el tamaño del individuo ni la aleatoriedad en las

diferencias, asumiendo que se distribuyen aleatoriamente y de una manera similar para cada observación.

- *Promedio de la suma del error cuadrático (MSE, por su sigla en inglés)*
  Este indicador realiza un promedio de la sumatoria de las diferencias cuadráticas entre los resultados generados por el individuo y los valores reales registrados por la serie de tiempo. Considerando la propuesta realizada en el Capítulo 2, es equivalente a la siguiente expresión:

$$Apt(I_i) = MSE(I_i) = \frac{\sum(I_i(\mathbf{x}, \boldsymbol{\theta}) - F(\mathbf{x}, \boldsymbol{\theta}))^2}{size(\mathbf{x})} \tag{3.2}$$

  Con $size(\mathbf{x})$ correspondiente al tamaño de la muestra. Este indicador es equivalente al $SSE/size(\mathbf{x})$, y al igual que el SSE suele ser utilizado en la comparación de los resultados entre modelos de predicción de series de tiempo por su facilidad de cálculo y resultado de error. Aunque se debe considerar que no toma en consideración el tamaño del individuo ni la aleatoriedad en las diferencias, asumiendo que se distribuyen aleatoriamente y de una manera similar para cada observación.

- *Desviación media absoluta (MAD, por su sigla en inglés)*
  Este indicador es considerado como una medida robusta de variabilidad para un conjunto de datos, el cual corresponde a la mediana de las diferencias absolutas entre los resultados generados por el individuo y los valores reales registrados por la serie de tiempo. Considerando la propuesta realizada en el Capítulo 2, es equivalente a la siguiente expresión:

$$Apt(I_i) = MAD(I_i) = median(|I_i(\mathbf{x}, \boldsymbol{\theta}) - F(\mathbf{x}, \boldsymbol{\theta})|) \tag{3.3}$$

  Con $median(.)$ correspondiente a la mediana del conjunto de datos dados por las diferencias absolutas entre los valores generados por I_i y las respectivas observaciones.

3.2.5 Pruebas estadísticas empleadas

Para el logro del objetivo específico del trabajo:

> *Implementar Implementar el algoritmo propuesto de programación genética, y validar sus bondades por medio de simulación contra series benchmark de la literatura de predicción de series de tiempo, comparando el error de predicción, la inclusión de los rezagos definidos, y la identificación de las componentes de los modelos predicción de series de tiempo.*

Se deben analizar los resultados obtenidos en términos del error de predicción, tamaño de los individuos, inclusión de los rezagos definidos y similitud con los modelos de predicción de series de tiempo presentes en la literatura, además del costo computacional de lograr la medida de error definida.

Por lo anterior, es necesario definir una serie de estadísticos a ser considerados para la correcta evaluación de las mejoras realizadas a partir de las modificaciones propuestas versus la versión original de GP, por lo que, en esta subsección se definirán aquellos utilizados por aspecto a evaluar.

3.2.6 Evaluación de la aptitud de los individuos

Tradicionalmente se ha evaluado como función de aptitud el error de predicción de los individuos seleccionados, lo que puede generar un crecimiento excesivo de los individuos sin aporte al valor de la aptitud,

también llamado bloat. Por ende, varios autores han realizado aportes en la metodología de análisis del error de predicción de los individuos durante el proceso de búsqueda del algoritmo de GP para una mejor comparación entre las posibles soluciones encontradas. A continuación se muestran las metodologías más representativas, las cuales serán tenidas en cuenta en la evaluación del algoritmo propuesto en el Capítulo 2:

- *Gráficos de caja de la aptitud de la población por generación*
  McPhee & Poli [24], proponen el uso de varias graficas de caja (Boxplot), entre las que se incluyen la distribución del valor de aptitud de cada uno de los individuos de la población por cada generación, en el cual, el eje horizontal corresponde a las generaciones y en el eje vertical al valor de aptitud.

- *Proporción de la población con mejora en el valor de aptitud por generación*
  McPhee & Poli [24], proponen el uso de del grafico de proporción de mejora de los individuos de la población entre generaciones para la evaluación de la capacidad del algoritmo de GP para generar nuevas soluciones en zonas del espacio de mayor interés, en el cual, el eje horizontal corresponde a las generaciones y en el eje vertical al a la proporción de individuos con mejor valor de aptitud, equivalente a: $prop_g = \sum\{1, si\ Apt(bInd_{g-1}) < Apt(I_i);\ 0, en\ caso\ contrario\}/n$, con $bInd$ vector de mejores individuos por generación.

- *Mejor individuo (en valor de aptitud) por generación*
  Hara et al. [22], Zhao et al. [23], Ono et al. [27], Karaboga et al. [40], Borges et al. [44], Costa & Pozo [53], Cagnoni et al. [54], y Aichour & Batouche [69]; plantean el uso de la gráfica de valor de aptitud (eje vertical) por generación (eje horizontal) para constrastar la mejora en el valor de aptitud de las modificaciones propuestas al algoritmo original de GP.

- *Promedio de la aptitud de los individuos por generación*
  Hoang et al. [8], Uy et al. [30], Imada & Ross [35], Tomassini et al. [38], Costa & Pozo [53], y Vanneschi & Cuccu [74]; validan la mejoras planteadas en el algoritmo de GP por medio de una gráfica del valor promedio de la aptitud de los individuos (en el eje vertical) versus el número de generaciones (en el eje horizontal).

- *Desviación estándar del valor de aptitud de la población por generación*
  Xiong & Wang [26], Agapitos et al. [33], Yuen & Leung [47], Cerny et al. [66], y Muntean et al. [67]; verifican los resultados de mejora del modelo planteado de GP por medio del análisis de los valores de desviación estándar, mínimo, máximo, media y mediana de la solución encontrada. Mientras que Fillon & Bartoli [49] solo utilizan los resultado de la desviación estándar para el análisis.

- *Relación de aproximación vs valor real*
  Gandomi et al. [49] propone el uso de la gráfica de comparación del valor del mejor individuo (eje vertical) versus el valor real (eje horizontal) para analizar la eficacia de las modificaciones planteadas al algoritmo de GP.

Realizando un análisis detallado de cada una de las metodologías planteadas, es posible agrupar en el grafico de Caja de la aptitud de la población por generación propuesto por McPhee & Poli [24] los graficos de: el Mejor individuo (en valor de aptitud) por generación [22, 23, 27, 40, 44, 53, 54, 69], en el cual el valor máximo de cada uno de las cajas (boxplot) del grafico corresponde al mejor individuo por cada una de las

generaciones; el Promedio de la aptitud de los individuos por generación [8, 30, 35, 38, 53, 74], en la cual dicho promedio es representado por la línea al interior del boxplot.

3.2.7 Evaluación del tamaño de los individuos

El crecimiento excesivo de los individuos sin aporte al valor de la aptitud, también llamado bloat, es uno de los principales problemas en la aplicación de la programación genética a la predicción de series temporales, debido a la degradación en la calidad de las soluciones sucesivamente en cada generación [81, 82].

Para afrontar este problema, en el Capítulo 2, se propusieron una serie de mejoras al algoritmo original de GP basadas en la eliminación de constantes en el conjunto de terminales por medio del uso exclusivo de bloques funcionales, la modificación de la función de aptitud empleada que castigara la complejidad de la solución (representada por el tamaño del individuo), y el uso del operador de simplificación.

Para evaluar las mejoras en el tamaño de los individuos durante el proceso de búsqueda, varios autores han propuesto técnicas de evaluación gráfica, las cuales son:

- *Promedio del tamaño de los individuos de la población por generación*
  Tomassini et al. [38], Kinzett et al. [63], y Zhang & Wong [64]; analizan la gráfica de tamaño promedio en nodos de la población (eje vertical) versus el tamaño de la población (eje horizontal), con el fin de determinar las mejoras en la disminución del tamaño de los individuos.

- *Número de nodos únicos por generación*
  Kinzett et al. [63] realiza una gráfica del conteo de nodos distintos de todos los individuos (eje vertical) por generación (eje horizontal) para identificar la efectividad del método de simplificación planteado.

- *Gráfico de cajas de la aptitud de la población por tamaño por generación*
  McDermott et al. [71] realiza una gráfica de aptitud y tamaño del mejor individuo (eje vertical) por generación (eje horizontal) para analizar las mejoras en el tamaño de los individuos manteniendo la capacidad de predecir la serie de tiempo. Se propone por ende, además de graficar al mejor individuo, graficar todos los individuos de la población por medio de un boxplot.

Teniendo en cuenta que un grafico de cajas (boxplot) contiene información tanto de los quartiles, el promedio, el minimo y el máximo de un conjunto de datos. Y que cada uno de los nodos del algoritmo propuesto GP-FB es único gracias al paso de simplificación (poda de árboles) implementado en la Subsección 2.2.10. Es posible agrupar los graficos presentados en uno único (complementario al Caja de la aptitud de la población por generación [24], analizado en la Subsección 3.2.6) en el que se grafique por medio de boxplots el número de nodos (tamaño) de los individuos de la población (eje vertical) para cada una de las generaciones (eje horizontal).

3.2.8 Evaluación del costo computacional

Para analizar el costo computacional, Tomassini et al. [38] analiza la gráfica de esfuerzo correspondiente al número de operaciones realizadas como aproximación a la medida de costo computacional (se suele reemplazar también por el tiempo de procesamiento) en el eje vertical, versus el valor de la aptitud en el eje horizontal. Lo anterior muestra una medida de capacidad de escalabilidad del algoritmo propuesto ante nuevas series de tiempo (complejidad y tamaño de la muestra).

Por otra parte, Ribeiro & Rosseti [121] realizan una comparación entre los distintos modelos por medio de la grafica Time-to-Target (tttplot), en la cual se en el eje vertical la probabilidad acumulada de que el

algoritmo encuentre una solución igual o superior al valor de aptitud dado. Mientras que el eje horizontal corresponde al tiempo (y en ciertos casos al número de generaciones) necesario para lograrlo.

3.3 Series de tiempo benchmark analizadas

Para el análisis de las capacidades de predicción del algoritmo propuesto de programación genética —en el que se incluyen las modificaciones propuestas en el Capítulo 2— fueron seleccionadas las series de tiempo AIRLINE, LYNX, POLLUTION, INTERNET y SUNSPOT, las cuales son consideradas bechmark en la literatura de predicción de series de tiempo, debido a sus propiedades estadísticas que incluyen: estacionalidades, tendencias y alta correlación de sus rezagos.

Cabe resaltar que cada serie fue comparada en la medida de error reportada por los mejores resultados suministrados en la literatura por otras técnicas de predicción de series de tiempo. La validación de la mejora entre el algoritmo propuesto en esta tesis GP-FB (genetic programming with functional blocks) y la propuesta metodológica presentada en la tesis de maestría (gpModel), se realizo por medio de la prueba estadistica de *Wilcoxon Rank-Sum Test* [122] en la que se compara la mediana de dos muestras relacionadas y determina si existen diferencias entre ellas.

Por otra parte, en las implementaciones del algoritmo original de GP, la propuesta realizada en la tesis de maestría gpModel [12] y el algoritmo propuesto en este trabajo GP-FB, se incluyeron las mejoras propuestas de rendimiento de la sección 2.2.14, entre las que se incluye la evaluación matricial de X y el uso de expresiones regulares para la evaluación del individuo. Lo anterior para que sean consistentes los resultados de la evaluación de rendimiento por medio de las graficas Time to Target [121].

3.3.1 Serie AIRLINE

La serie AIRLINE corresponde al número de pasajeros transportados mensualmente al exterior por una aerolínea entre Enero de 1949 (ENE1949) y Diciembre de 1960 (DIC1960); esta serie presenta una tendencia creciente y un patrón cíclico de periodicidad anual, los cuales exhiben un comportamiento no lineal.

Ésta serie ha sido identificada como la serie G por Box & Jenkins [13] para sus análisis de modelos ARIMA. Posteriormente, Faraway & Chatfield [123] construyeron una red neuronal MLP con 4 neuronas en la capa oculta y una sola en la capa de salida, con entradas $\mathbf{x} = \{x_{t-1}, \dots, x_{t-12}\}$, que significó la optimización de 21 parámetros en la ecuación resultante. Para hallar aquellos valores que permitían aproximar mejor el modelo a la serie, utilizaron el algoritmo backpropagation incremental con 10.000 corridas cuyos errores de aproximación están consignados en la Tabla 3-1.

Por otro lado, De Menezes & Nikolaev [51] realizaron un análisis de los modelos de programación genética y redes neuronales polinomiales (PGP), el cual corresponde a una hibridación entre la GP y las redes neuronales, considerando cada sub-árbol como un polinomio, el cual puede ser optimizado localmente y cuyo error de aproximación puede ser sumado como un error global del individuo con una cierta ponderación.

Adicionalmente, Velásquez et al. [17] realizaron un análisis de la capacidad de predicción de las Máquinas de Vectores de Soporte (SVM, por su sigla en inglés) comparándolo contra los resultados de las MLP, estimando para ello varias SVM con los mismos rezagos utilizados por Faraway & Chatfield [123], para los cuales fueron calculados los respectivos errores de entrenamiento, siendo menores que el generado por ARIMA entre un 82% y 100%. En general, las modelos de SVM en comparación a los MLP presentan una mejora del 23% en el error de aproximación, y de un 100% en el de predicción.

También, en la tesis de maestría de Martínez [12] se analizo la capacidad de predicción del algoritmo original de programación genética (GP) comparándolo contra los resultados de las modificaciones propuestas (inclusión de bloques funcionales en los terminales de los individuos), seleccionando los modelos de menor error de entrenamiento y predicción, en el cual se pudo apreciar una mejora del 87% en el SSE de entrenamiento y 98% en el SSE de validación entre el modelo original de GP y el modelo propuesto gpModel.

En este trabajo se realizó un análisis de la serie de tiempo por medio del algoritmo de GP modificado (GP-FB), incluyendo las propuestas presentadas en el Capítulo 2, en el cual los primeros 120 datos son utilizados para la estimación del modelo y los 12 restantes para su pronóstico. Dicha serie es transformada usando la función logaritmo natural [16, 17], la cual se puede apreciar en la Figura 3-1, función de error SSE, algoritmo de optimización OPTIM [109] y de profundización RGNOUD [120], aplicando las pruebas propuestas en las subsecciones 3.2.6 a 3.2.8, las cuales pueden ser apreciadas en la Figura 3-2 para el algoritmo original de GP [3], en la Figura 3-3 para el algoritmo de gpModel [12] y en la Figura 3-4 para la propuesta de este trabajo GP-FB. Adicionalmente, a partir del análisis realizado por medio del índice de auto correlación parcial no lineal propuesto por Martinez [111], se determinó el uso de $p = 13$ número de rezagos (consistente con los resultados reportados en la literatura de predicción de series de tiempo), $q = p - 1$ número de diferencias, y $d = 1$ nivel de diferenciación.

Los resultados, tanto de entrenamiento como de pronostico son presentados en la Tabla 3-1, en la cual son comparados los resultados obtenidos en la literatura con modelos ARIMA, MLP, SVM, ANN, el modelo original de GP identificado por GP, el modelo modificado propuesto en la tesis de maestría por Martínez [12] identificado por gpModel, y el algoritmo modificado (propuesto en el Capítulo 2) identificado por GP-FB. En dicha tabla es posible apreciar que el GP-FB posee un SSE menor que los reportados por los demás modelos, con una diferencia del 87% con respecto al modelo ARIMA. Adicionalmente, en comparación con los modelos SVM, el GP-FB presenta un menor error de entrenamiento entre 22% y el 84%, y un menor error de predicción entre el 50% y 92%. Igualmente, el GP-FB presenta un mejor comportamiento en entrenamiento y predición que los modelos ANN (mínima mejora: 46,2%; máxima mejora: 98,2%), MPL (mínima mejora: 46.2%; máxima mejora: 93,9%), ARIMA (mejora del 87%), GP reporta una mejora del 89,4%. Por otro lado, comparado GP-FB con el algoritmo modificado gpModel también reporta una mejora del 17,6% en el error de aproximación.

En la Figura 3-1 se puede apreciar tanto la serie original AIRLINE (transformada por logaritmo natural) y el mejor modelo GP-FB (de acuerdo con la corrida realizada con los parámetros antes mencionados), para los datos de entrenamiento y de pronóstico, la cual muestra que el modelo realmente encierra el comportamiento de los datos y no solo los memoriza (problema común de falta de generalización en la predicción de series de tiempo); esto también puede ser evidenciado en la similitud entre los errores de entrenamiento y predicción (Tabla 3-1).

En la Figura 3-2 se muestran los criterios de evaluación del modelo de predicción generado por GP (tradicional) para la serie AIRLINE: a) Gráfico de boxplot de la aptitud de la población por generación; b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación; c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación; d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación; e) Gráfico de la relación de aproximación vs valor real. En los cuales se puede apreciar que los modelos generados por GP exhiben una alta distribución del error llegando a 3.800 (a), repercutiendo en los valores de la desviación estándar, centrándose en 0,5 (d); aunque mantiene estable el tamaño de los individuos entre 1 y 17 nodos (b); debido principalmente a la aplicación del operador de cruce tradicional. Sin una estabilización clara de las mejoras de la proporción de individuos mejores (menor medida de error – mayor valor de Aptitud) entre generaciones entre un 28% y un 30% (c) e influyendo claramente en su aproximación con una alta dispersión de los puntos alrededor de la recta de 45 grados (e).

Tabla 3-1: Resultados de entrenamiento y pronóstico entre modelos de predicción para la serie AIRLINE.

Modelo	Rezagos	Neuronas ocultas	C	ε	σ	Entrenamiento SSE	Predicción SSE
GP-FB*	**1 – 13**					**0.14**	**0,01**
gpModel**	**1 – 13**					**0.17**	**0,01**
ARIMA[1]	1, 12, 13					1,08	0,43
MLP con mejor ajuste[1]	1 – 13					0,26	1,12
MLP con mejor pronóstico[1]	1, 12					2,30	0,34
SVM-1[1]			268	0,05	1,0	0,88	0,13
SVM-2[1]			363	0,04	10,5	0,20	0,00
SVM-3[1]			348	0,02	1,0	0,34	0,01
SVM-4[1]			346	0,02	3,8	0,35	0,03
SVM-5[1]			878	0,04	0,6	0,18	0,20
SVM-6[1]			518	0,03	20,0	0,21	0,02
SVM-7[1]			343	0,09	6,7	0,21	0,13
SVM-8[1]			562	0,09	3,0	0,36	0,05
SVM-9[1]			562	0,08	2,3	0,26	0,04
SVM-10[1]			557	0,08	2,4	0,26	0,05
GP[2]	1-13					1,32	0,8
ANN-1[2]	1, 2, 3, 4	2				7,74	1,03
ANN-2[2]	1 – 13	2				0,73	0,71
ANN-3[2]	1 – 13	4				0,26	1,12
ANN-4[2]	1, 12	2				2,30	0,34
ANN-5[2]	1, 12	4				2,16	0,44
ANN-6[2]	1, 12	10				1,77	0,59
ANN-7[2]	1, 2, 12	2				2,17	0,29
ANN-8[2]	1, 2, 12	4				1,91	1,03
ANN-9[2]	1, 2, 12, 13	2				0,99	0,52
ANN-10[2]	1, 2, 12, 13	4				0,81	0,52
ANN-11[2]	1, 12, 13	1				1,18	0,50
ANN-12[2]	1, 12, 13	2				1,03	0,50
ANN-13[2]	1, 12, 13	4				0,84	0,62

* Generación propia; Test Wilcox p-value = 0.002 con respecto a gpModel.

** Martínez [12]

[1]Velásquez et al. [17]

[2]Ghiassi et al. [16]

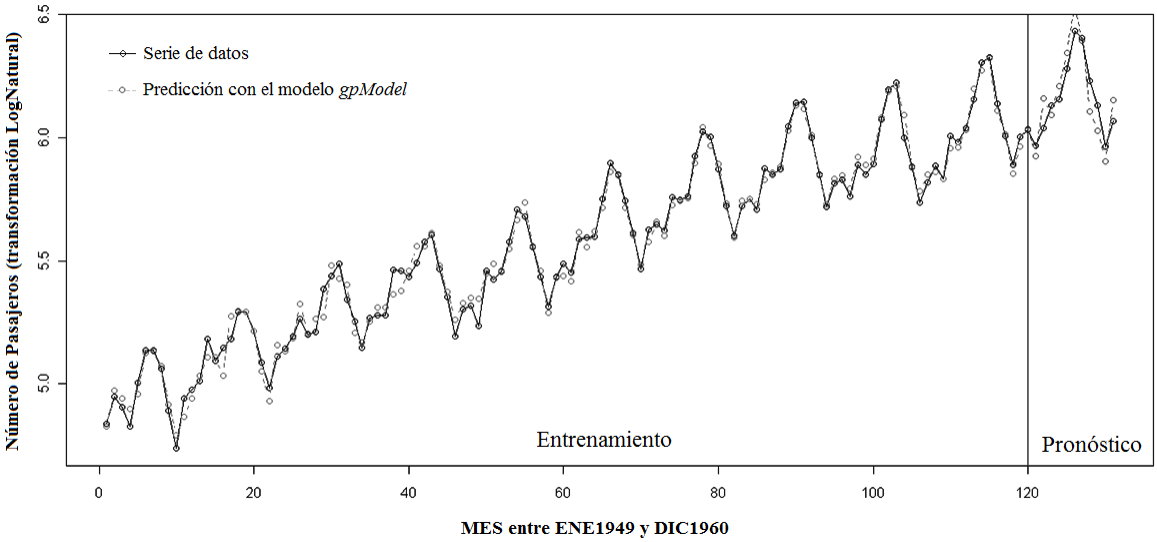


Figura 3-1: Resultados de entrenamiento y pronóstico del modelo de predicción para la serie AIRLINE por medio de la metodología propuesta.

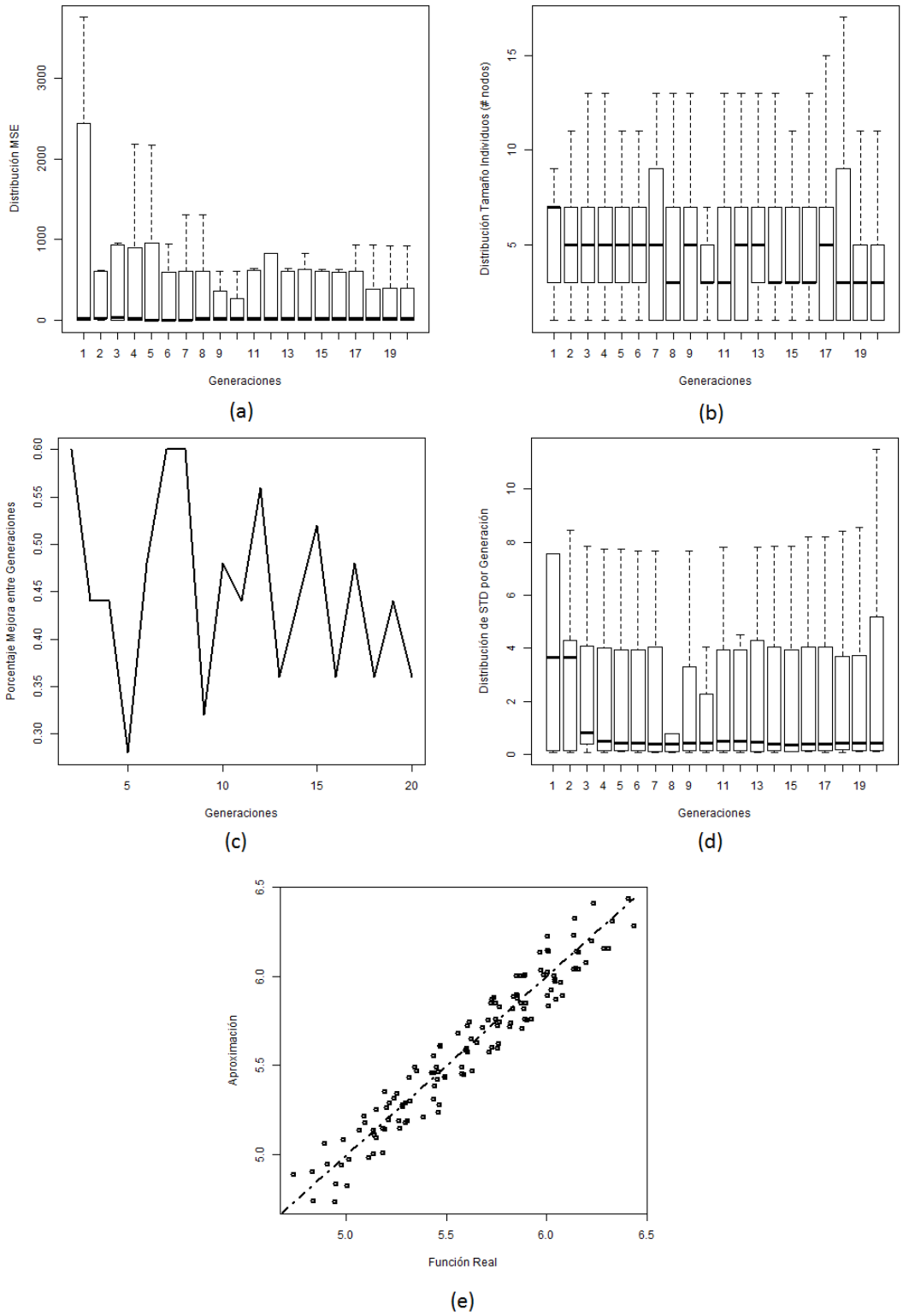


Figura 3-2: Criterios de evaluación del modelo de predicción generado por GP (tradicional)para la serie AIRLINE: a) Gráfico de boxplot de la aptitud de la población por generación; b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación; c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación; d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación; e) Gráfico de la relación de aproximación vs valor real.

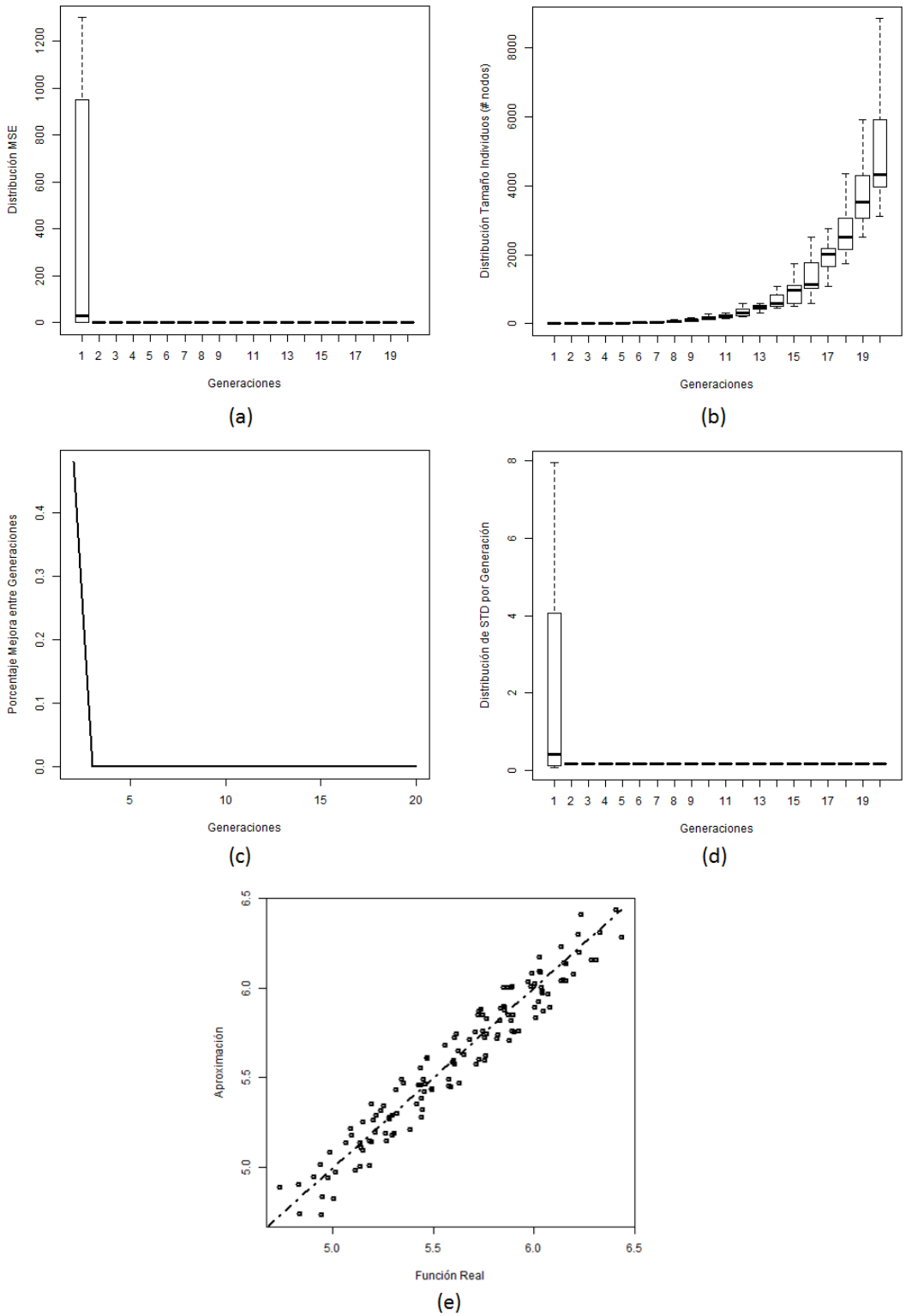
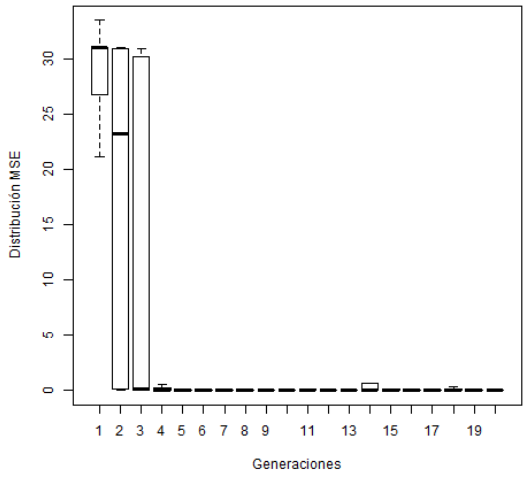
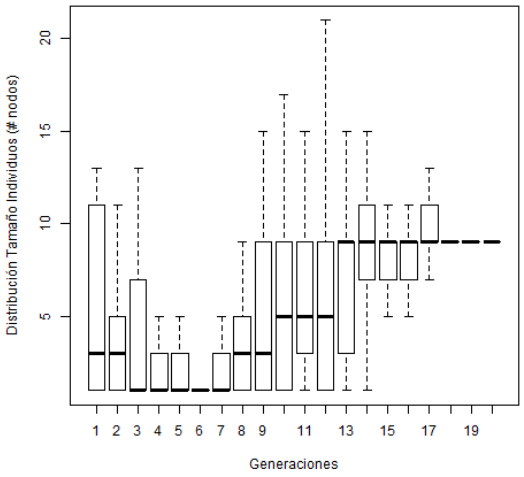


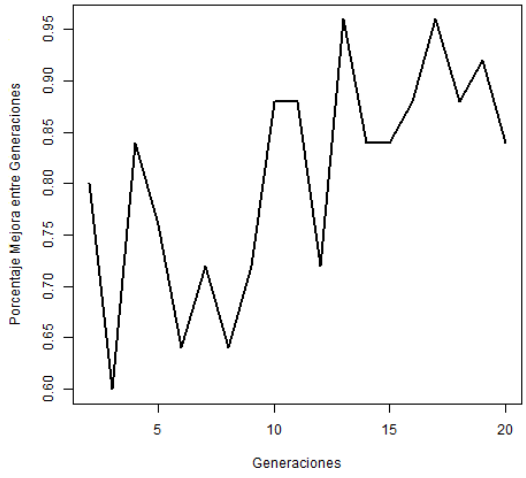
Figura 3-3: Criterios de evaluación del modelo de predicción generado por gpModel para la serie AIRLINE: a) Gráfico de boxplot de la aptitud de la población por generación; b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación; c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación; d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación; e) Gráfico de la relación de aproximación vs valor real.


Distribución MSE
Generaciones

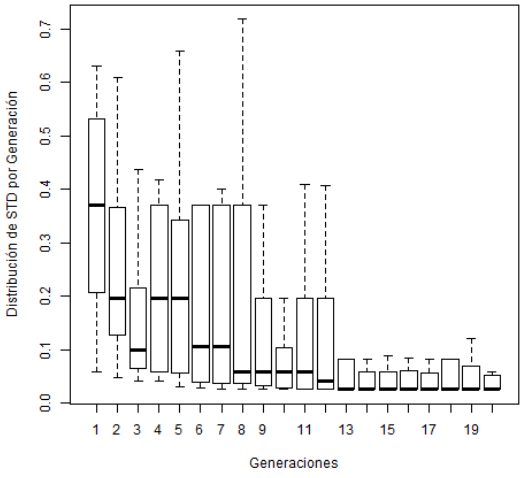
(a)


Distribución Tamaño Individuos (# nodos)
Generaciones

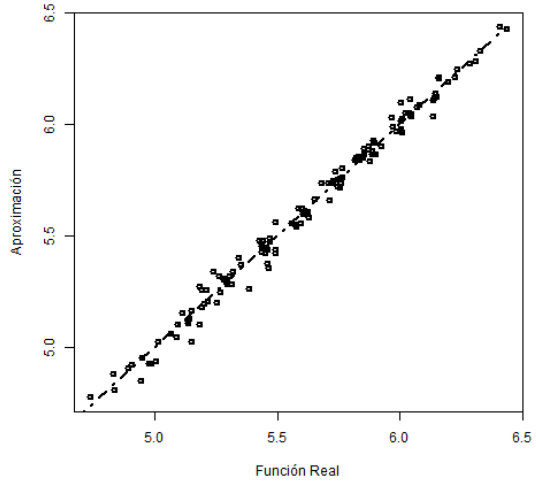
(b)


Porcentaje Mejora entre Generaciones
Generaciones

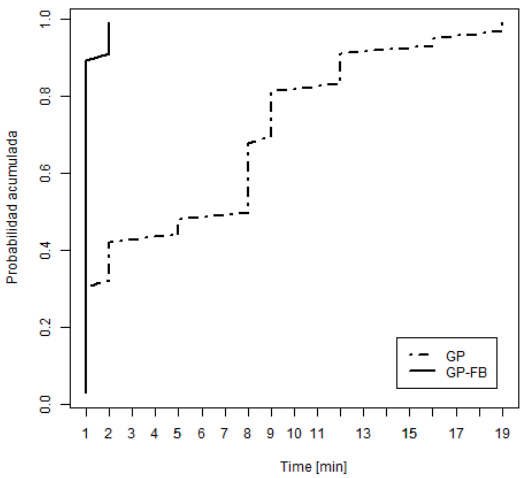
(c)


Distribución de STD por Generación
Generaciones

(d)


Aproximación
Función Real

(e)


Probabilidad acumulada
GP
GP-FB
Time [min]

(f)

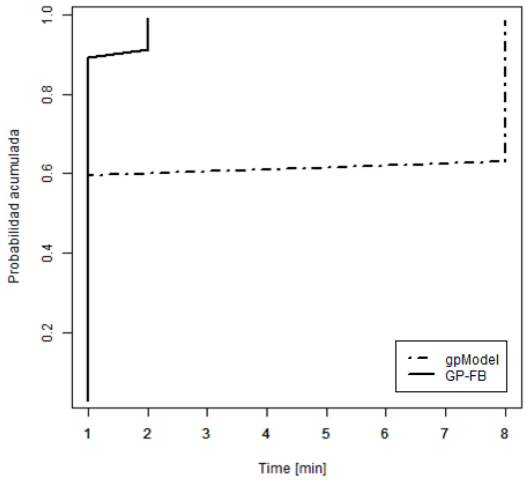


Figura 3-4: Criterios de evaluación del modelo de predicción generado por GP-FB para la serie AIRLINE: a) Gráfico de boxplot de la aptitud de la población por generación; b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación; c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación; d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación; e) Gráfico de la relación de aproximación vs valor real; f) Gráfica Time-to-Target (tttplot), entre los modelos GP (tradicional) y GP-FB; g) Gráfica Time-to-Target (tttplot), entre los modelos gpModel y GP-FB.

Si bien, los cambios propuestos en la tesis de maestría [12] generan mejoras en la predicción de series de tiempo con GP, aún se muestran falencias en los criterios de aceptación de los modelos resultantes. Lo anterior se evidencia en la Figura 3-3, en la que se muestran los criterios de evaluación del modelo de predicción generado por gpModel para la serie AIRLINE; en a) se muestra que el error continua siendo alto desde la primera generación (hasta 1.300), aunque disminuye en las siguientes generaciones, asociado principalmente al uso de bloques funcionales en los terminales (gráfico de boxplot de la aptitud de la población por generación); lo que conlleva a que se presente valores dispersos en la desviación estándar (entre 0.5 y 8), como se muestran en d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación; Sin embargo, en b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación, muestra un crecimiento desmedido en el número de nodos (tendencia exponencial) entre 1 y 9.000 nodos, debido principalmente a la metodología de cruce con suma sin poda. Por otra parte, en c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación, se muestra un decrecimiento a partir de la tercera generación (desde un 47%), en línea con los demás graficos, en el que se muestra que una vez ha alcanzado un individuo con menor medida de error, las modificaciones realizadas por los operadores genéticos en la población no impactan realmente en el resultado (generación de bloat), y cuyos resultados se reglejan en e) Gráfico de la relación de aproximación vs valor real, en el cual se aprecia la dispersión de puntos alrededor de la recta de 45 grados.

Por otra parte, en la Figura 3-4 se muestran los resultados del algoritmo propuesto con las modificaciones planteadas en el Capítulo 2, en el que claramente se aprecia una disminución en la medida de error desde la primera generación (SSE = 30), cuyas mejoras son replicadas a los demás individuos de la población, llegando a un nivel de SSE = 0,14 general en la segunda generación, como se puede ver en a) Gráfico de boxplot de la aptitud de la población por generación. Al igual que gpModel, GP-FB tendría un crecimiento desmedido en los individuos, pero es efectivamente controlado por medio del operador de simplificación (Subsección 2.2.10) por lo que en b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación, se muestra un crecimiento lineal con un promedio de 9 nodos y un máximo de 21 nodos. Debido a un nivel de

diversificación alto en los individuos, se puede apreciar un alto porcentaje (entre el 60% y el 95) en c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación. Todo lo anterior influye positivamente en d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación, en el que tienen tendencia decreciente (mucho menor a GP y gpModel) y media cercana a cero; y e) Gráfico de la relación de aproximación vs valor real, en el que se muestra de nuevo (al igual que la Figura 3-1), la capacidad del algoritmo de encerrar el comportamiento de la serie. Igualmente, en el análisis de convergencia en f) Gráfica Time-to-Target (tttplot), entre los modelos GP (tradicional) y GP-FB, y g) Gráfica Time-to-Target (tttplot), entre los modelos gpModel y GP-FB, se concluye que GP-FB tiene una alta probabilidad de converger a un valor de aptitud (o error) dado en muy poco tiempo, para el caso particular de la serie AIRLINE en 2 minutos, en contraposición a GP que lo hace en 19 minutos y gpModel en 8 minutos.

3.3.2 Serie LYNX

La serie LYNX corresponde al número de linces canadienses atrapados por año en el distrito del río Mckenzie del norte de Canadá entre los años 1821 y 1934 (114 observaciones). En la literatura se suele modelar la serie transformándola por medio del logaritmo base 10 y tomando los primeros 100 datos para el entrenamiento y los 14 restantes para la predicción.

Esta serie ha sido ampliamente estudiada por Campbell & Walker [124], quienes muestran a partir de gráficos que la serie posee una periodicidad de 10 años con irregularidades en amplitud. Zhang [125], por su parte, reporta los resultados del uso de modelos ARIMA y ANN con estructura $7 \times 5 \times 1$ y un modelo híbrido, siendo este último mejor en términos del MSE y MAD de aproximación que los dos anteriores.

También Velásquez et al. [17] han estudiado esta serie por medio de SVM, los anteriores resultados pueden ser apreciados en la Tabla 3-2, en la cual son comparados contra el modelo a partir de GP.

En este trabajo se aplicó tanto el modelo de GP original [3] y el algoritmo propuesto (ver Capítulo 2) GP-FB, en el cual, los primeros 100 datos son utilizados para la estimación del modelo y los 14 restantes para su pronóstico. Dicha serie es transformada usando la función logaritmo base 10 [124], la cual se puede apreciar en la Figura 3-5 (línea continua). Para la ejecución del algoritmo fueron utilizados 20 individuos de población inicial (3 niveles de profundidad y 7 terminales a lo máximo), un número máximo de 10 generaciones, función de error SSE, algoritmo de optimización OPTIM [109] y de profundización RGNOUD [120] (Subsección 3.2).

Los resultados, tanto de entrenamiento como de pronóstico son presentados en la Tabla 3-2, en la cual son comparados resultados obtenidos en la literatura con modelos ARIMA, MLP y SVM, y el modelo original de GP identificado por GP, los cuales son comparables a los obtenidos por la metodología propuesta (GP-FB).

En la Figura 3-5 se puede apreciar tanto la serie original LYNX (transformada por logaritmo base 10) y el mejor modelo GP-FB (de acuerdo con la corrida realizada con los parámetros antes mencionados), para los datos de entrenamiento (lado izquierdo de la línea vertical) y de pronóstico (línea discontinua y círculos vacíos), la cual muestra que el modelo realmente encierra el comportamiento de los datos y no solo los memoriza (problema común de falta de generalización en la predicción de series de tiempo); esto también puede ser evidenciado en la similitud entre los errores de entrenamiento y predicción en SSE y una menor volatilidad MAD (Tabla 3-2).

Tabla 3-2: Resultados de entrenamiento y pronóstico entre modelos de predicción para la serie LYNX.

Modelo	Rezagos	C	ε	σ	Entrenamiento SSE (MAD)	Predicción SSE (MAD)
GP-FB*	**1 – 10**				**0,020 (0,101)**	**0,013 (0,092)**
gpModel**	**1 – 10**				**0,028 (0,115)**	**0,017 (0,102)**
ARIMA[1]	N/D				N/D	0,021 (0,112)
GP[1]	1-10				6,81 (19,24)	1.98 (3,15)
MLP[1]	N/D				N/D	0,021 (0,112)
Híbrido[1] (MLP -ARIMA)[1]	1 – 7				N/D	0,017 (0,104)
SVM-1[1]	1 – 6	261	0,2	3,7	0,036 (0,161)	0,021 (0,111)
SVM-2[1]	1 – 7	731	0,2	4.8	0,034 (0,160)	0,019 (0,121)
SVM-3[1]	1 – 8	477	0,2	5,0	0,034 (0,155)	0,020 (0,125)
SVM-4[1]	1 – 9	444	0,2	2,8	0,026 (0,140)	0,036 (1,163)
SVM-5[1]	1 – 10	549	0,2	4,4	0,031 (0,157)	0,035 (0,163)
SVM-6[1]	1 - 3, 8 – 10	587	0,2	4,0	0,034 (0,152)	0,015 (0,087)
SVM-71	1-4,8-10	468	0,2	6,9	0,038 (0,160)	0,021 (0,122)

* Generación propia; Test Wilcox p-value = 0.002 con respecto a gpModel.

** Martínez [12]

[1]Velásquez et al. [17]

El GP-FB presenta un SSE de entrenamiento menor que los reportados por los demás modelos SVM (para los ARIMA y MLP no se registran datos). En términos del SSE de predicción, el GP-FB es mejor que los datos obtenidos por lo SVM, en el cual posee una mejora de al menos un 23,1% (hasta un 47,4%) y entre un 27.9% y 37.3% con respecto al MAD. GP-FB comparado con los modelos ARIMA, el error de predicción SSE mejora un 38% (con un MAD menor en un 18%); con respecto a los modelos MLP en los cuales se registra un error de predicción SSE menor entre 23,5% y 38,1%, con una mejora en el MAD entre el 11,5% y el 17,9%. Comparado GP-FB con el algoritmo original de GP también reporta una mejora del 99,7% con respecto al error de entrenamiento y del 99,3% al de predicción. Adicionalmente, comparando GP-FB con la modificación propuesta en la tesis de maestría gpModel [12], se reporta una mejora del 28,6% en el error de entrenamiento SSE y del 23,5% en el error de predicción SSE.

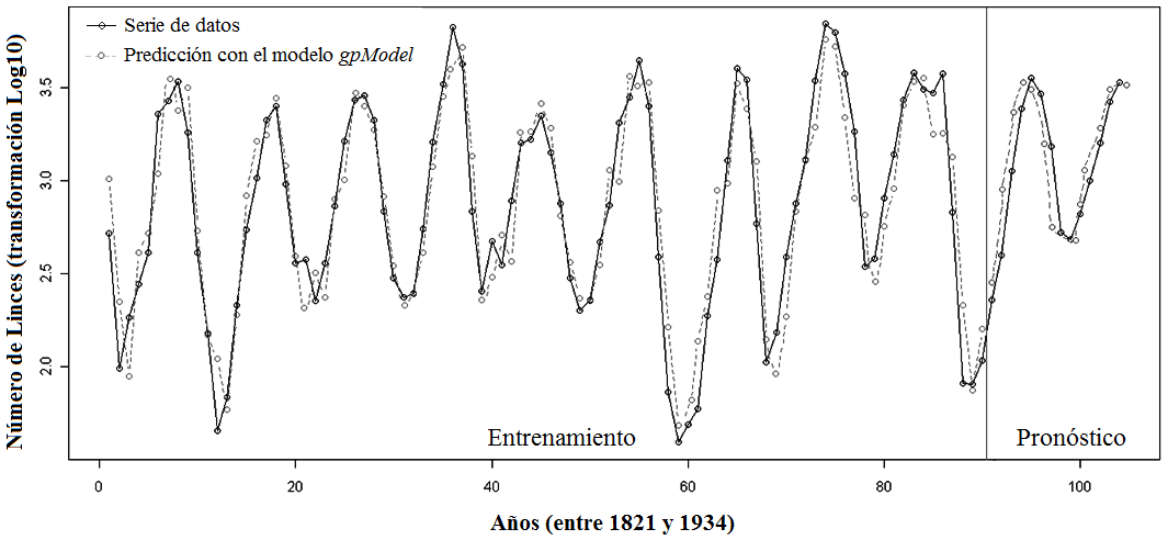


Figura 3-5: Resultados de entrenamiento y pronóstico del modelo de predicción para la serie LYNX por medio de la metodología propuesta.

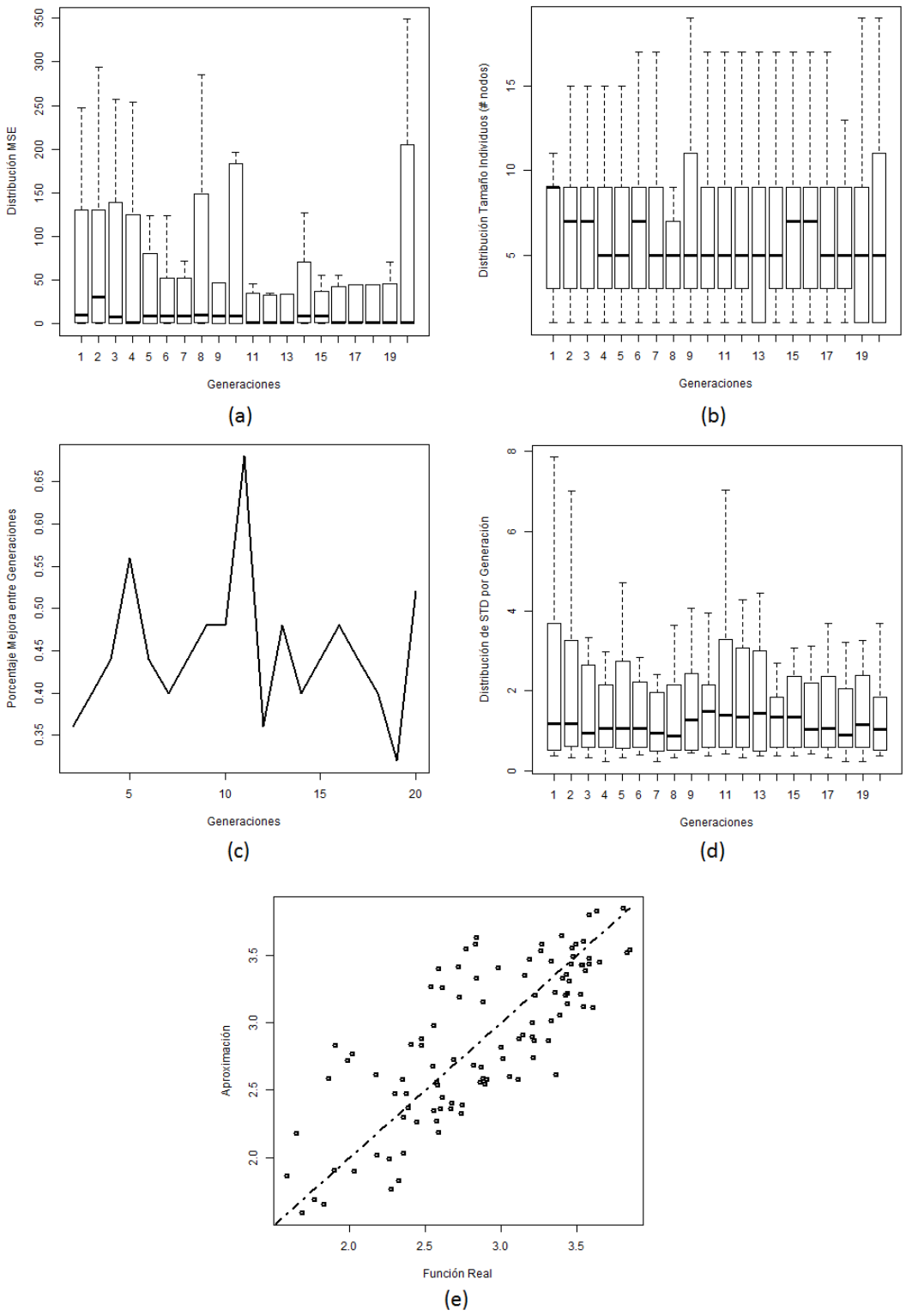


Figura 3-6: Criterios de evaluación del modelo de predicción generado por GP (tradicional) para la serie LYNX: a) Gráfico de boxplot de la aptitud de la población por generación; b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación; c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación; d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación; e) Gráfico de la relación de aproximación vs valor real.

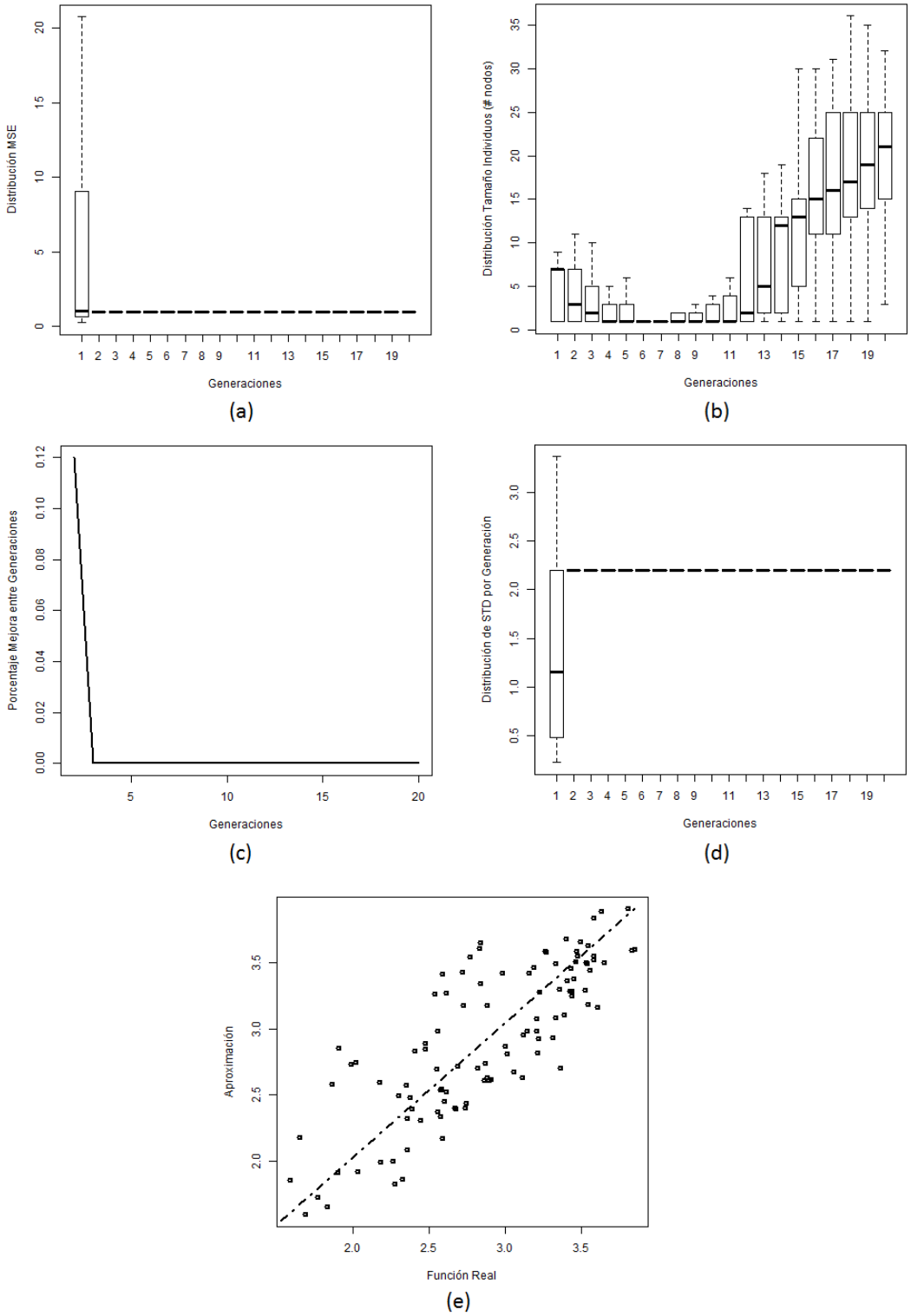
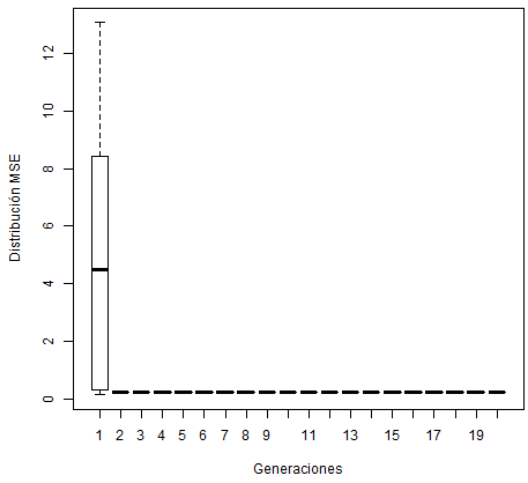
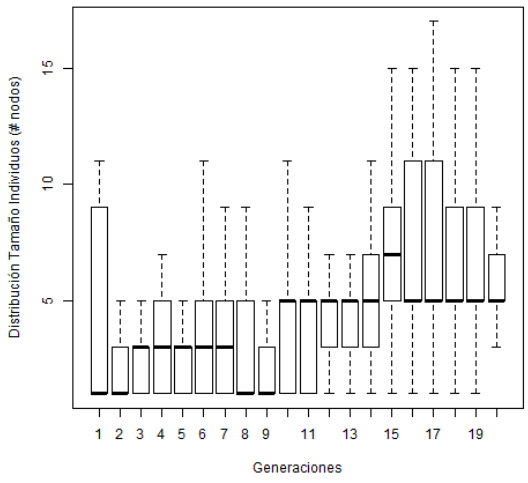


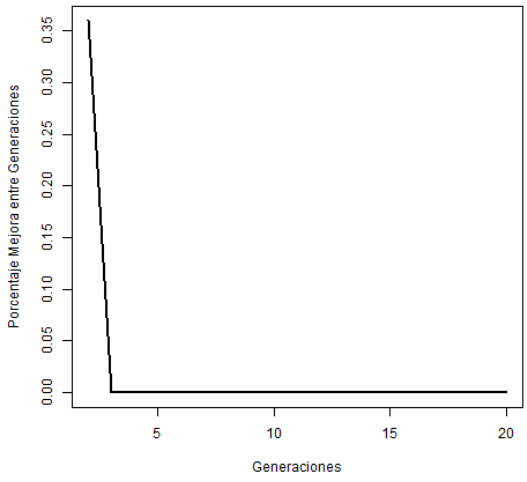
Figura 3-7: Criterios de evaluación del modelo de predicción generado por gpModel para la serie LYNX: a) Gráfico de boxplot de la aptitud de la población por generación; b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación; c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación; d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación; e) Gráfico de la relación de aproximación vs valor real.


Distribución MSE
0 2 4 6 8 10 12
1 2 3 4 5 6 7 8 9 11 13 15 17 19
Generaciones

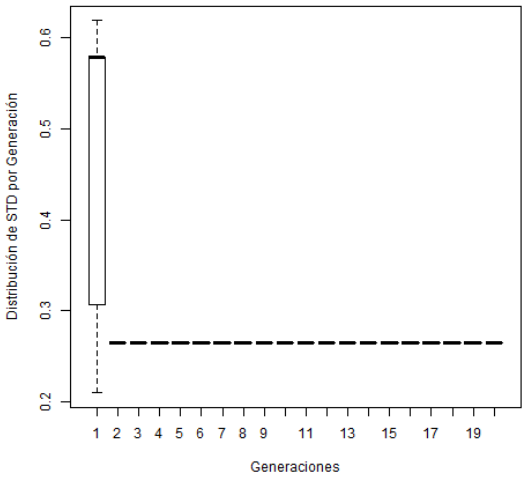
(a)


Distribución Tamaño Individuos (# nodos)
5 10 15
1 2 3 4 5 6 7 8 9 11 13 15 17 19
Generaciones

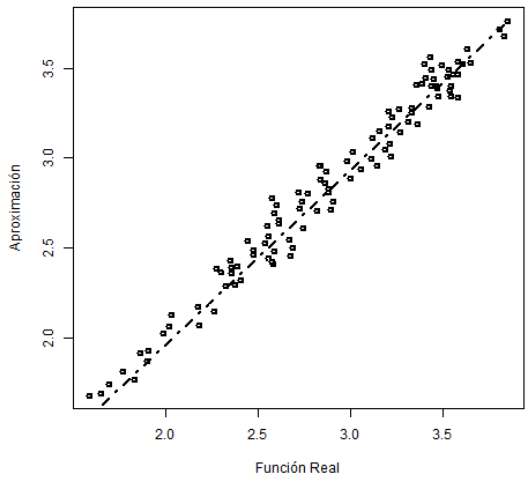
(b)


Porcentaje Mejora entre Generaciones
0.00 0.05 0.10 0.15 0.20 0.25 0.30 0.35
5 10 15 20
Generaciones

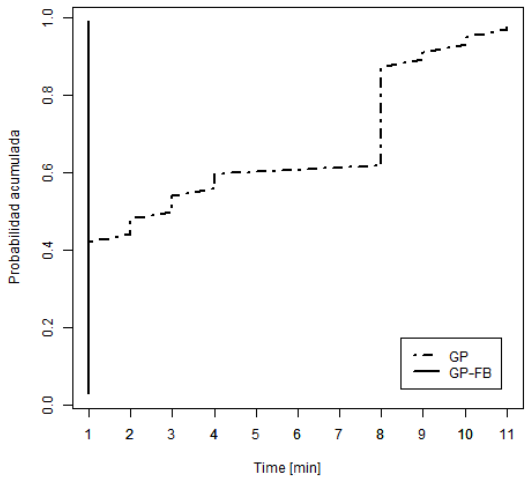
(c)


Distribución de STD por Generación
0.2 0.3 0.4 0.5 0.6
1 2 3 4 5 6 7 8 9 11 13 15 17 19
Generaciones

(d)


Aproximación
2.0 2.5 3.0 3.5
2.0 2.5 3.0 3.5
Función Real

(e)


Probabilidad acumulada
0.0 0.2 0.4 0.6 0.8 1.0
1 2 3 4 5 6 7 8 9 10 11
Time [min]
GP
GP-FB

(f)

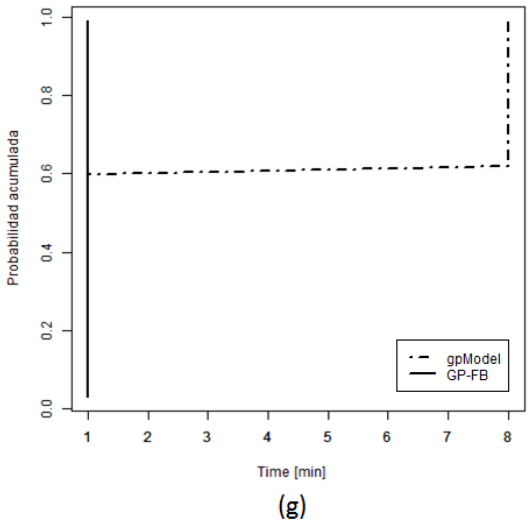


Figura 3-8: Criterios de evaluación del modelo de predicción generado por GP-FB para la serie LYNX: a) Gráfico de boxplot de la aptitud de la población por generación; b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación; c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación; d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación; e) Gráfico de la relación de aproximación vs valor real; f) Gráfica Time-to-Target (tttplot), entre los modelos GP (tradicional) y GP-FB; g)) Gráfica Time-to-Target (tttplot), entre los modelos gpTool y GP-FB.

En la Figura 3-6 se muestran los criterios de evaluación del modelo de predicción generado por GP (tradicional) para la serie LYNX: a) Gráfico de boxplot de la aptitud de la población por generación; b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación; c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación; d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación; e) Gráfico de la relación de aproximación vs valor real. En los cuales se puede apreciar que los modelos generados por GP exhiben una alta distribución del error (a) llegando inclusive a un SSE = 350 y repercutiendo en los valores de la desviación estándar (d) cercano al 2, aunque mantiene estable el tamaño de los individuos (b) entre 5 y 10 nodos en promedio; debido principalmente a la aplicación del operador de cruce tradicional. Sin una estabilización clara de las mejoras de la proporción de individuos mejores (menor medida de error – mayor valor de Aptitud, oscilando entre el 35% y el 70%) entre generaciones (c) e influyendo claramente en su aproximación (e) el cual esta claramente muy disperso del valor esperado (línea recta con ángulo de 45 grados).

Si bien, los cambios propuestos en la tesis de maestría [12] generan mejoras en la modelación con GP, aún se muestran falencias en los criterios de aceptación de los modelos resultantes. Lo anterior se evidencia en la Figura 3-7, en la que se muestran los criterios de evaluación del modelo de predicción generado por gpModel para la serie LYNX; en a) se muestra que el comienza en SSE = 21 en la primera generación, disminuyendo en las siguientes asociado al uso de los bloques funcionales (gráfico de boxplot de la aptitud de la población por generación); lo que conlleva a una disminución de la desviación estándar centrada en 2,2, como se muestran en d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación; Sin embargo, en b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación, muestra un crecimiento (tendiendo a exponencial) en el número de nodos debido principalmente a la metodología de cruce con suma sin poda, llegando a un máximo de 35 nodos. Por otra parte, en c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación, se muestra un decrecimiento a partir de la tercera generación (comenzando en el 12%), en línea con los demás graficos, en el que se muestra que una vez ha alcanzado un individuo con menor medida de error, las modificaciones realizadas por los operadores genéticos

en la población no impactan realmente en el resultado (generación de bloat), y cuyos resultados se reglejan en e) Gráfico de la relación de aproximación vs valor real, en cuyo caso es mejor que GP, pero que sigue alejado del valor objetivo correspondiente a la recta con ángulo de 45 grados.

Por otra parte, en la Figura 3-8 se muestran los resultados del algoritmo propuesto con las modificaciones planteadas en el Capítulo 2, en el que claramente se aprecia una disminución en la medida de error desde la primera generación (centrado en SSE = 4.5), cuyas mejoras son replicadas a los demás individuos de la población, llegando a un nivel general en la tercera generación (SSE = 0.020), como se puede ver en a) Gráfico de boxplot de la aptitud de la población por generación. Al igual que gpModel, GP-FB tendría un crecimiento desmedido en los individuos, pero es efectivamente controlado por medio del operador de simplificación (Subsección 2.2.10) por lo que en b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación, se muestra un crecimiento lineal con un promedio de 5 nodos y un máximo de 16 nodos. Debido principalmente a la rápida convergencia de los individuos, c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación, tiende a disminuir con tendencia a cero desde un 35% inicial de mejora entre las dos primeras generaciones. Todo lo anterior influye positivamente en d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación, en el que tienen tendencia decreciente (mucho menor a GP y gpModel, arrancando en 1,2) y media cercana a cero; y e) Gráfico de la relación de aproximación vs valor real, en el que se muestra de nuevo (al igual que la Figura 3-5), la capacidad del algoritmo de encerrar el comportamiento de la serie (centrada sobre la recta con ángulo de 45 grados). Igualmente, en el análisis de convergencia en f) Gráfica Time-to-Target (tttplot), entre los modelos GP (tradicional) y GP-FB, y g) Gráfica Time-to-Target (tttplot), entre los modelos gpModel y GP-FB, se concluye que GP-FB tiene una alta probabilidad de converger a un valor de aptitud (o error) dado en muy poco tiempo, para el caso particular de la serie LYNX en el primer minuto, en contraposición a GP que lo hace en 11 minutos y gpModel en 8 minutos.

3.3.3 Serie POLLUTION

La serie POLLUTION contiene los datos de la cantidad de despachos mensuales de un equipo de polución en miles de francos franceses entre enero de 1986 (ENE1986) y octubre de 1996 (OCT1996), con un total de 130 observaciones. La serie a ser pronosticada corresponde al logaritmo natural de los datos originales; las primeras 106 observaciones son usadas para la estimación de los modelos, mientras que las 24 restantes son usadas para su pronóstico.

En la Figura 3-9 se puede apreciar la gráfica de su comportamiento, el cual exhibe una tendencia que fluctúa en el tiempo con una magnitud variable demostrada por el Bnon estacionario en la varianza. Ésta fue estudiada en sus inicios por Makridakis et al. [126] quienes indicaron que un modelo ARIMA de orden $(2,1,1) \times (1,0,0)_{12}$ es adecuado para representar su dinámica.

Guiassi et al. [16] realizan un análisis comparativo entre los modelos DAN2, los NN y ARIMA, considerando para ello las primeras 106 observaciones de entrenamiento y las restantes 12 y 24 (separadamente) para validación del pronóstico. En general, reporta un mejor comportamiento al considerar 15 rezagos con mejoras del 40% con respecto a los modelos ARIMA.

Por otro lado, Velásquez et al. [17] después de realizar un análisis comparativo entre los modelos MLP, ARIMA y SVM, concluyen que los modelos SVM predicen de mejor manera la serie POLLUTION de acuerdo con los errores de predicción tanto a 12 como a 24 meses, los cuales son inferiores a los reportados por los otros dos métodos.

En este trabajo se aplicó tanto el modelo de GP original [3] identificado por GP y el algoritmo modificado (metodología propuesta en el Capítulo 2) identificado por GP-FB, en el cual los primeros 106 datos son utilizados para la estimación del modelo y los 24 restantes para su pronóstico. Dicha serie es transformada usando la función logaritmo base 10 [124], la cual se puede apreciar en la Figura 3-3 (línea continua). Para la ejecución del algoritmo modificado fueron utilizados 50 individuos de población inicial (3 niveles de profundidad y 7 terminales a lo máximo), un número máximo de 15 generaciones, función de error SSE,

algoritmo de optimización OPTIM [109] y de profundización RGNOUD [120]. Cabe recordar que el error de pronóstico fue tomado tanto para 12 como 24 meses por separado para ser consistentes con los resultados presentados en la literatura.

En la Figura 3-9 se puede apreciar la serie original POLLUTION (transformada por logaritmo natural) y el mejor modelo GP-FB (de acuerdo con la corrida realizada con los parámetros antes mencionados), para los datos de entrenamiento (lado izquierdo de la línea vertical) y de pronóstico (línea discontinua y círculos vacíos), la cual muestra que el modelo realmente encierra el comportamiento de los datos y no solo los memoriza (problema común de falta de generalización en la predicción de series de tiempo); esto también puede ser evidenciado en la similitud entre los errores de entrenamiento y predicción en SSE y una menor volatilidad MAD (Tabla 3-3).

Los resultados, tanto de entrenamiento como de pronóstico que se muestran en la Tabla 3-3, en la cual se comparan resultados obtenidos en la literatura con modelos ARIMA, MLP, SVM, ANN, GP, algoritmo modificado gpModel propuesto en la tesis de maestría [12], y el algoritmo propuesto en esta tesis GP-FB. El modelo GP-FB tanto para entrenamiento como pronóstico registra un MSE menor a los reportados por los demás modelos. Es de resaltar que tanto para los modelos ARIMA (mejora del 65,4%), MLP (mejora del 66,7%) como para las SVM (mínima mejora: 63,3%; máxima mejora: 65,4%) cuyas estructuras de generación están incluidas en los bloques funcionales utilizados; con respecto a los modelos ANN se presentan mejoras de un 66,7%, y una menor aleatoriedad en el error medido bajo el criterio MAD (45,2%). Las anteriores mejoras en las medidas de error también son apreciables en la predicción, en la cual a 1 año van desde el 20% para el modelo ARIMA (aunque en el criterio de MAD mejora en un 81,5%) hasta el 89,6% para los modelos MLP y ANN (con una mejora del 93,7 % en el MAD); siendo aún más notoria a un horizonte de 2 años, en cuyo caso presenta una mejora en el MSE del 92,5 % con respecto al modelo ARIMA (mejora del 93,7% en el MAD), 86,3% contra el MLP y ANN (mejora del 91,1% en el MAD), y van del -52,4% al -75% en los SVM (mejora del 85,7% al 88,7% en el MAD). Comparando GP-FB con el algoritmo original también reporta una mejora del 79,5% en el error de aproximación y del 75,9% en el de predicción a 1 año y del 90,7% en el de 2 años; y con respecto a la modificación gpModel una mejora del 21,7% en el error de aproximación y del 25,9% en el de predicción a 1 año y del 75,9% en el de 2 años. Es d resaltar que para todos los resultados reportados, el modelo propuesto GP-FB posee un MAD inferior, que implica que es más ajustado a la serie de tiempo tanto en error como estructura general.

Tabla 3-3: Resultados de entrenamiento y pronóstico entre modelos de predicción para la serie POLLUTION.

Modelo	Rezagos	C	ε	σ	Entrenamiento MSE (MAD)	Predicción MSE (MAD) 1 año	Predicción MSE (MAD) 2 años
GP-FB*	**1 - 15**				**0,018 (0,103)**	**0,020 (0,025)**	**0,077 (0,101)**
gpModel**	**1 - 15**				**0,023 (0,131)**	**0,027 (0,034)**	**0,083 (0,124)**
ARIMA[1]	1 - 3, 12 - 15				0,052 (0,181)	0,025 (0,135)	0,268 (0,395)
GP[1]	1-15				0,088 (0,190)	0.083 (0.141)	0.214 (0.281)
MLP[1]	1 - 12				0,054 (0,188)	0,193 (0,394)	0,146 (0,334)
SVM-1[1]	1 - 3, 12 - 15	220	0,08	32,0	0,052 (0,176)	0,031 (0,159)	0,042 (0,175)
SVM-2[1]	1 - 12	1.100	0,10	28,0	0,049 (0,174)	0,039 (0,159)	0,080 (0,221)
ANN[1]	1 - 12				0,054 (0,188)	0,193 (0,394)	0,146 (0,334)

* Generación propia; Test Wilcox p-value = 0.023 con respecto a gpModel.

** Martínez [12]

[1]Velásquez et al. [17]

En la Figura 3-10 se muestran los criterios de evaluación del modelo de predicción generado por GP (tradicional) para la serie POLLUTION: a) Gráfico de boxplot de la aptitud de la población por generación; b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación; c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación; d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación; e) Gráfico de la relación de aproximación vs valor

real. En los cuales se puede apreciar que los modelos generados por GP exhiben una alta distribución del error (a) llegando inclusive a un SSE = 1.500 y repercutiendo en los valores de la desviación estándar (d) cercano al 2 (con un máximo en 19), aunque mantiene estable el tamaño de los individuos (b) entre 3 y 5 nodos en promedio (y un máximo de 15 nodos); debido principalmente a la aplicación del operador de cruce tradicional. Sin una estabilización clara de las mejoras de la proporción de individuos mejores (menor medida de error – mayor valor de Aptitud, oscilando entre el 30% y el 55%) entre generaciones (c) e influyendo claramente en su aproximación (e) el cual está claramente disperso del valor esperado (línea recta con ángulo de 45 grados).

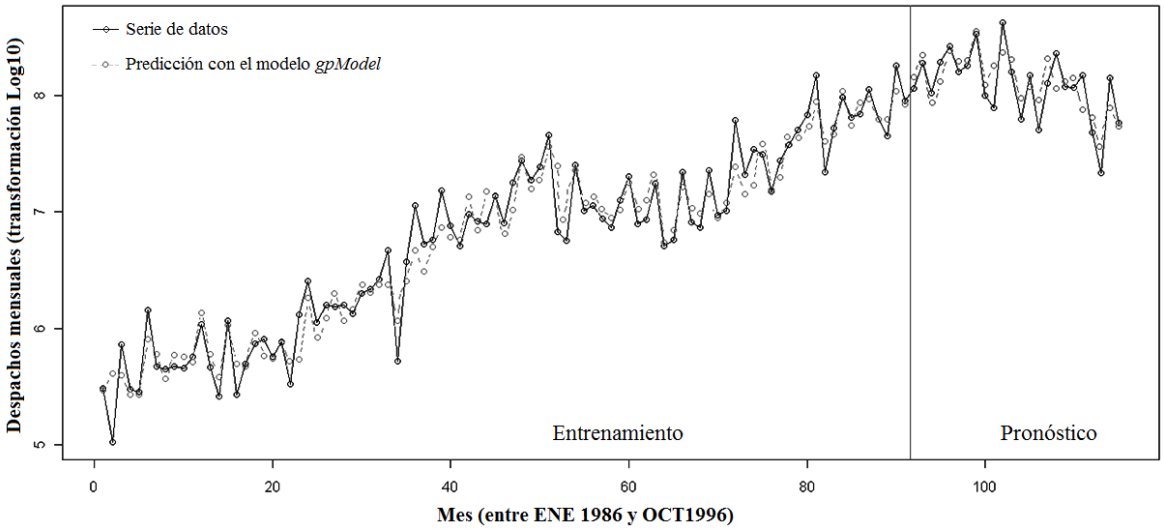


Figura 3-9: Resultados de entrenamiento y pronóstico del modelo de predicción para la serie POLLUTION por medio de la metodología propuesta.

Si bien los cambios propuestos en la tesis de maestría [12] generan mejoras en la modelación con GP, aún se muestran falencias en los criterios de aceptación de los modelos resultantes. Lo anterior se evidencia en la Figura 3-11, en la que se muestran los criterios de evaluación del modelo de predicción generado por gpModel para la serie POLLUTION; en a) se muestra que comienza con un SSE = 40 en promedio en la primera generación, estabilizándose en SSE = 200; lo que conlleva a una disminución de la desviación estándar cercana al 1 (en promedio) en la primera generación y centrada en 21 en las demás, como se muestran en d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación; Sin embargo, en b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación, muestra un crecimiento (exponencial) en el número de nodos debido principalmente a la metodología de cruce con suma sin poda, llegando hasta 35 nodos como máximo (y sin una clara mejora a la medida de error). Por otra parte, en c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación, se muestra un decrecimiento a partir de la tercera generación (desde el 12%), en línea con los demás graficos, en el que se muestra que una vez ha alcanzado un individuo con menor medida de error, las modificaciones realizadas por los operadores genéticos en la población no impactan realmente en el resultado (generación de bloat), y cuyos resultados se reglejan en e) Gráfico de la relación de aproximación vs valor real, en cuyo caso es mejor que GP, pero que sigue alejado del valor objetivo correspondiente a la recta con ángulo de 45 grados.

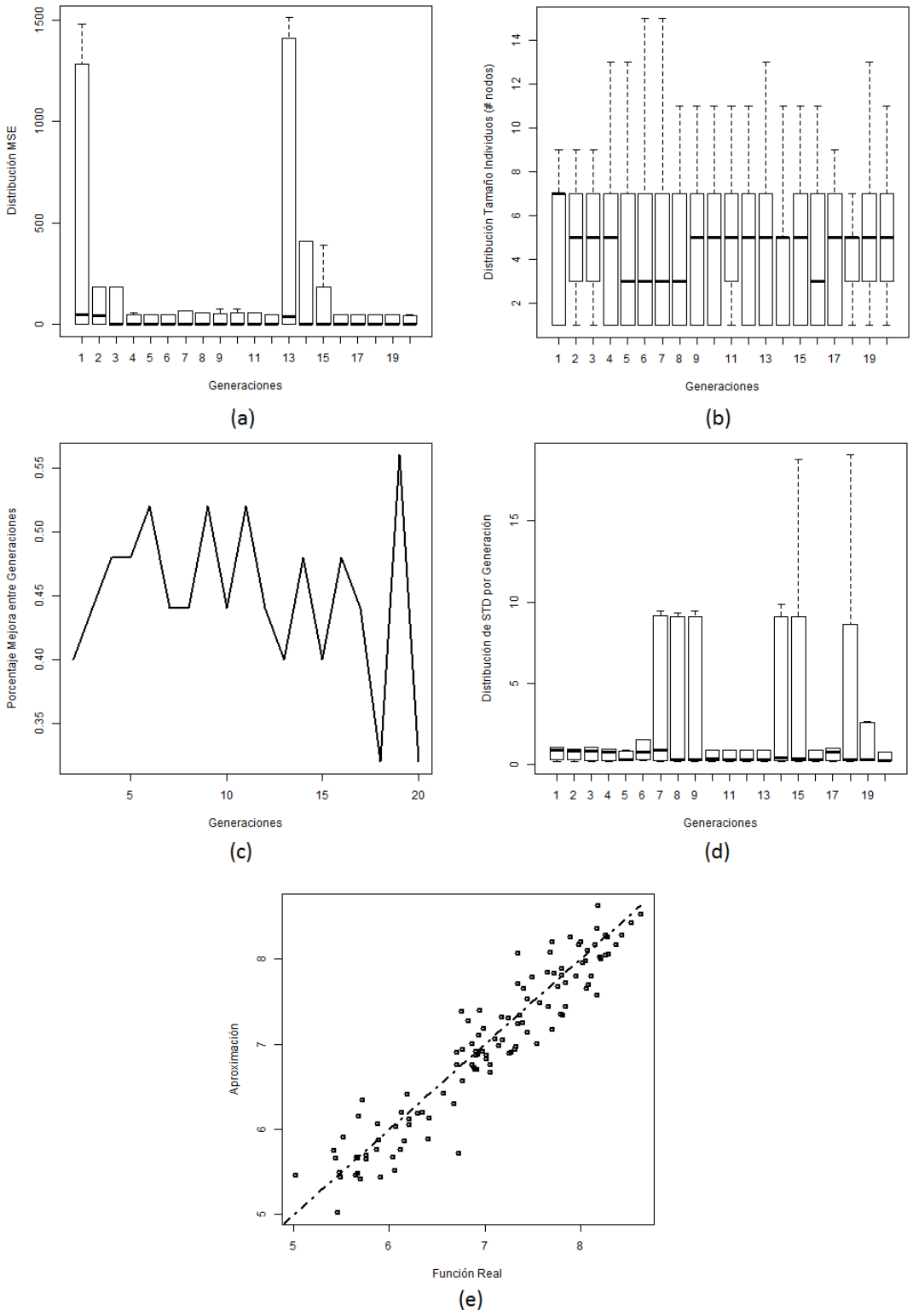


Figura 3-10: Criterios de evaluación del modelo de predicción generado por GP (tradicional) para la serie POLLUTION: a) Gráfico de boxplot de la aptitud de la población por generación; b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación; c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación; d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación; e) Gráfico de la relación de aproximación vs valor real.

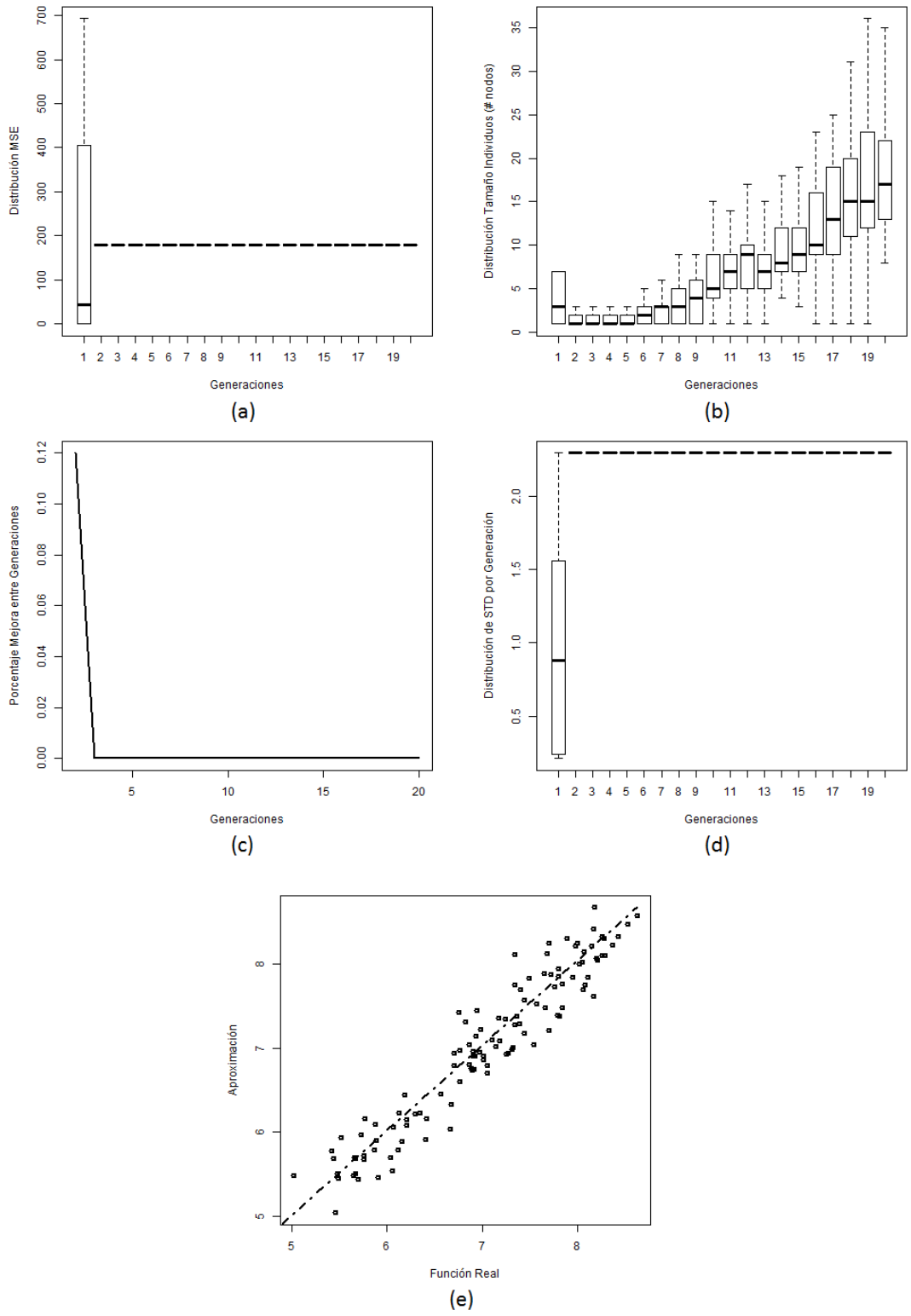
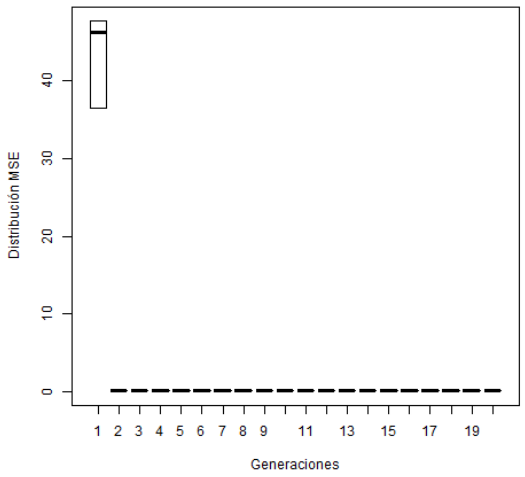
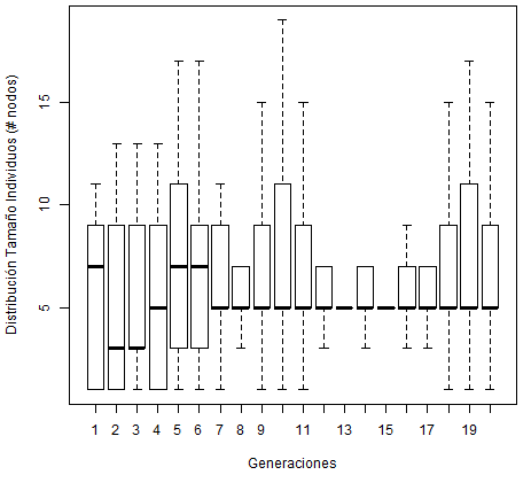


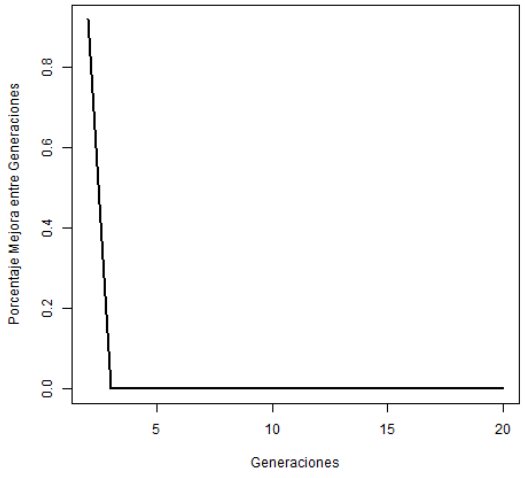
Figura 3-11: Criterios de evaluación del modelo de predicción generado por gpModel para la serie POLLUTION: a) Gráfico de boxplot de la aptitud de la población por generación; b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación; c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación; d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación; e) Gráfico de la relación de aproximación vs valor real.


Distribución MSE
0
10
20
30
40
1 2 3 4 5 6 7 8 9 11 13 15 17 19
Generaciones

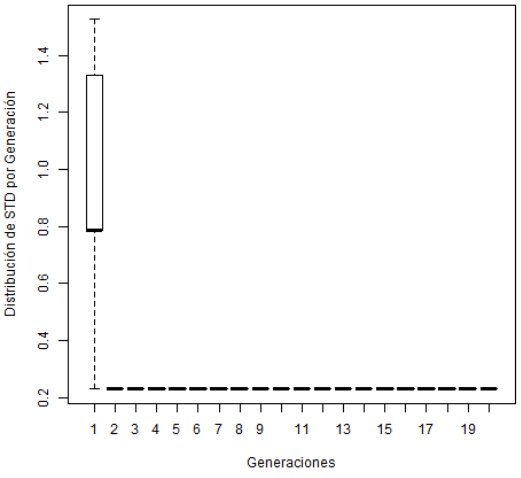
(a)


Distribución Tamaño Individuos (# nodos)
5
10
15
1 2 3 4 5 6 7 8 9 11 13 15 17 19
Generaciones

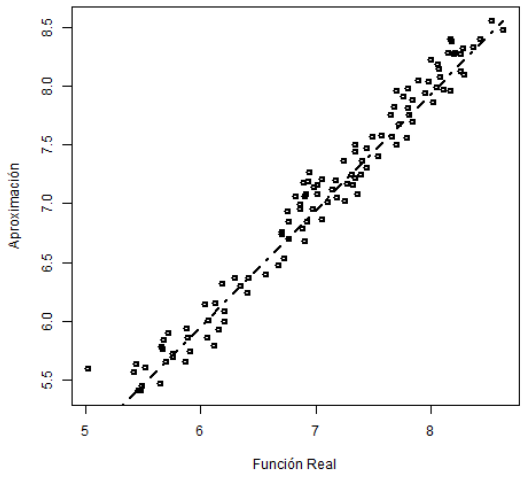
(b)


Porcentaje Mejora entre Generaciones
0.0
0.2
0.4
0.6
0.8
5 10 15 20
Generaciones

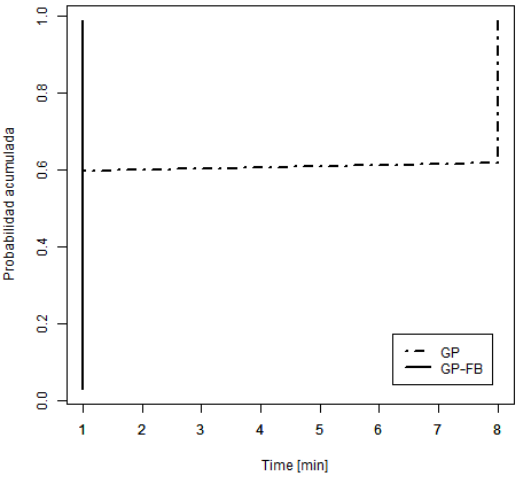
(c)


Distribución de STD por Generación
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1 2 3 4 5 6 7 8 9 11 13 15 17 19
Generaciones

(d)


Aproximación
5.5
6.0
6.5
7.0
7.5
8.0
8.5
5 6 7 8
Función Real

(e)


Probabilidad acumulada
0.0
0.2
0.4
0.6
0.8
1.0
1 2 3 4 5 6 7 8
Time [min]
GP
GP-FB

(f)

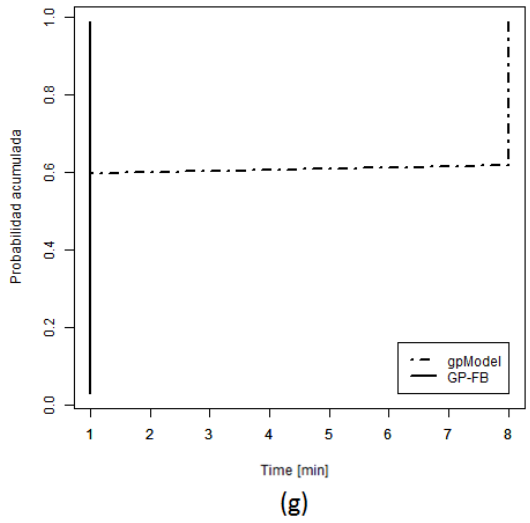


Figura 3-12: Criterios de evaluación del modelo de predicción generado por GP-FB para la serie POLLUTION: a) Gráfico de boxplot de la aptitud de la población por generación; b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación; c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación; d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación; e) Gráfico de la relación de aproximación vs valor real; f) Gráfica Time-to-Target (tttplot), entre los modelos GP (tradicional) y GP-FB; g)) Gráfica Time-to-Target (tttplot), entre los modelos gpTool y GP-FB.

Por otra parte, en la Figura 3-12 se muestran los resultados del algoritmo propuesto con las modificaciones planteadas en el Capítulo 2, en el que claramente se aprecia una disminución en la medida de error desde la primera generación (centrado en SSE = 46 equivalente a MSE = 0,433), cuyas mejoras son replicadas a los demás individuos de la población, llegando a un nivel general en la segunda generación (MSE = 0.018), como se puede ver en a) Gráfico de boxplot de la aptitud de la población por generación. Al igual que gpModel, GP-FB tendría un crecimiento desmedido en los individuos, pero es efectivamente controlado por medio del operador de simplificación (Subsección 2.2.10) por lo que en b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación, muestra un crecimiento lineal con un promedio de 5 nodos y un máximo de 19 nodos. Debido principalmente a la rápida convergencia de los individuos, c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación, tiende a disminuir con tendencia a cero desde un 87% inicial de mejora entre las dos primeras generaciones. Todo lo anterior influye positivamente en d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación, en el que tienen tendencia decreciente desde 0,8 a (mucho menor a GP y gpModel) una media cercana a 0,2; y e) Gráfico de la relación de aproximación vs valor real, en el que se muestra de nuevo (al igual que la Figura 3-9), la capacidad del algoritmo de encerrar el comportamiento de la serie (centrada sobre la recta con ángulo de 45 grados). Igualmente, en el análisis de convergencia en f) Gráfica Time-to-Target (tttplot), entre los modelos GP (tradicional) y GP-FB, y g) Gráfica Time-to-Target (tttplot), entre los modelos gpModel y GP-FB, se concluye que GP-FB tiene una alta probabilidad de converger a un valor de aptitud (o error) dado en muy poco tiempo, para el caso particular de la serie POLLUTION en el primer, en contraposición a GP que lo hace en 8 minutos y gpModel tambien 8 minutos.

3.3.4 Serie INTERNET

La serie INTERNET corresponde a la cantidad de usuarios que acceden a un servidor de internet por minuto durante 100 minutos consecutivos. Los primeros 80 datos son usados para la estimación del modelo, y los 20 restantes para su predicción. Se modela la serie original sin ningún tipo de transformación, su grafica se puede observar en la Figura 3-13, la cual exhibe el comportamiento no lineal de la serie sin presencia de ciclos aparentes y varios mínimos locales.

Esta serie ha sido estudiada por Makridakis et al. [126] quienes indican que los datos corresponden a un proceso no estacionario, en cuyo caso realizaron análisis gráficos y de correlación de los distintos rezagos concluyendo que es posible su descripción a partir de modelos ARIMA. Adicionalmente, luego de experimentar con series $ARIMA(p,1,q)$, sugiere que aquel de orden $(3,1,0)$ sería el más adecuado para describir su dinámica en función del MSE.

Ghiassi et al. [16] estudiaron esta serie a través del uso de DAN2 con igual partición de datos y medida de error MSE, los resultados obtenidos comparados contra los modelos ARIMA, sugieren que DAN2 posee una capacidad superior de aproximación con mejoras del 71% y 52% tanto en entrenamiento como pronóstico respectivamente.

Adicionalmente, Velásquez et al. [17] analizaron la serie por medio de los SVM, comparando los resultados de entrenamiento y pronóstico contra los modelos MLP y ARIMA, concluyendo que los modelos SVM no son adecuados para esta serie de datos, debido a que no logran llegar a una mejor alternativa que los modelos ARIMA Y MLP (en la Tabla 3-4 fue incluido el mejor resultado registrado en términos del MSE de entrenamiento y predicción de los modelos SVM).

En este trabajo se aplicó tanto el modelo de GP original [3] como el algoritmo modificado GP-FB (de acuerdo a la metodología propuesta en el Capítulo 2) en el cual los primeros 80 datos son utilizados para la estimación del modelo y los 20 restantes para su pronóstico. Para la ejecución del algoritmo modificado fueron utilizados 100 individuos de población inicial (3 niveles de profundidad y 7 terminales a lo máximo), un número máximo de 20 generaciones, función de error SSE, algoritmo de optimización OPTIM [109] y de profundización RGNOUD [120] (ver Subsección 3.2).

En la Figura 3-13 se puede apreciar tanto la serie original INTERNET y el mejor modelo GP-FB (de acuerdo con la corrida realizada con los parámetros antes mencionados), para los datos de entrenamiento (lado izquierdo de la línea vertical) y de pronóstico (línea discontinua y círculos vacíos), la cual muestra que el modelo realmente encierra el comportamiento de los datos y no solo los memoriza (problema común de falta de generalización en la predicción de series de tiempo); esto también puede ser evidenciado en la similitud entre los errores de entrenamiento y predicción en SSE y una menor volatilidad MAD (Tabla 3-4).

Tabla 3-4: Resultados de entrenamiento y pronóstico entre modelos de predicción para la serie INTERNET.

Modelo	Rezagos	C	ε	σ	Entrenamiento MSE (MAD)	Predicción MSE (MAD)
GP-FB*	**1 – 4**				**3,7 (1,30)**	**4,98 (1,81)**
gpModel**	**1 – 4**				**4,1 (1,60)**	**5,07 (1,92)**
ARIMA[1]	1 – 4				9,76 (2,42)	8,11 (2,23)
GP[1]	1 – 4				10,51 (2,16)	27,67 (3,22)
MLP[1]	1 – 4				7,00 (2,10)	9,25 (2,25)
SVM[1]	1 – 4	5.900	4,7	140	9,30 (2,50)	16,58 (3,44)

* Generación propia; Test Wilcox p-value = 0.002 con respecto a gpModel.

** Martínez [12]

[1]Velásquez et al. [17]

Los resultados, tanto de entrenamiento como de pronóstico son mostrados en la Tabla 3-4, en la cual se comparan resultados obtenidos en la literatura con modelos ARIMA, MLP, SVM, el modelo GP original

identificado por GP, el modelo modificado propuesto en la tesis de maestría [12] gpModel, y el algoritmo propuesto GP-FB. El modelo propuesto GP-FB presenta un MSE de entrenamiento menor que los reportados por los demás modelos, principalmente comprado con los resultados del modelo ARIMA (mejora del 62,1%), MLP (mejora del 47,1%) y las SVM (mejora del 60,2%); adicionalmente presenta una menor aleatoriedad en el error medido bajo el criterio MAD (mínima mejora: 38,1%; máxima mejora: 48%); éstos resultados también son apreciables en la predicción, en los cuales presenta disminución del error MSE que van desde el 38,6% para el modelo ARIMA (mejora en el MAD del 18,8%) hasta el 70% para el modelo SVM (con una mejora del 47,4 % en el MAD) pasando por el modelo MLP con una mejora del 46,2% (con una mejora del 19,6% en el MAD). Comparando GP-FB con el algoritmo original GP también reporta una mejora del 64,8% en el error de aproximación y del 82% en el de predicción; y con respecto al modelo gpModel una mejora del 9,8% en el error de aproximación y del 1,8% en el de predicción.

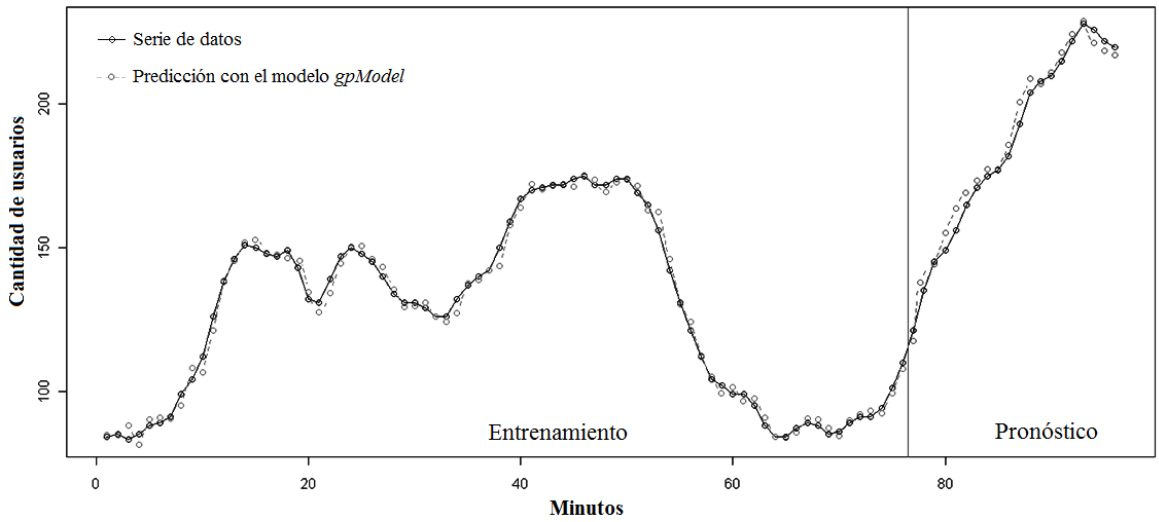


Figura 3-13: Resultados de entrenamiento y pronóstico del modelo de predicción para la serie INTERNET por medio de la metodología propuesta.

En la Figura 3-14 se muestran los criterios de evaluación del modelo de predicción generado por GP (tradicional) para la serie INTERNET: a) Gráfico de boxplot de la aptitud de la población por generación; b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación; c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación; d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación; e) Gráfico de la relación de aproximación vs valor real. En los cuales se puede apreciar que los modelos generados por GP exhiben una alta distribución del error (a) llegando inclusive a un SSE = 150.000 y repercutiendo en los valores de la desviación estándar (d) cercano al 32 con un máximo promedio en 60, aunque mantiene estable el tamaño de los individuos (b) entre 1 y 5 nodos en promedio, debido principalmente a la aplicación del operador de cruce tradicional, sin una estabilización clara de las mejoras de la proporción de individuos mejores (menor medida de error – mayor valor de Aptitud, oscilando entre el 25% y el 53%) entre generaciones (c) e influyendo claramente en su aproximación (e) el cual está claramente disperso (aunque sin outliers y de amplitud controlada) del valor esperado (línea recta con ángulo de 45 grados).

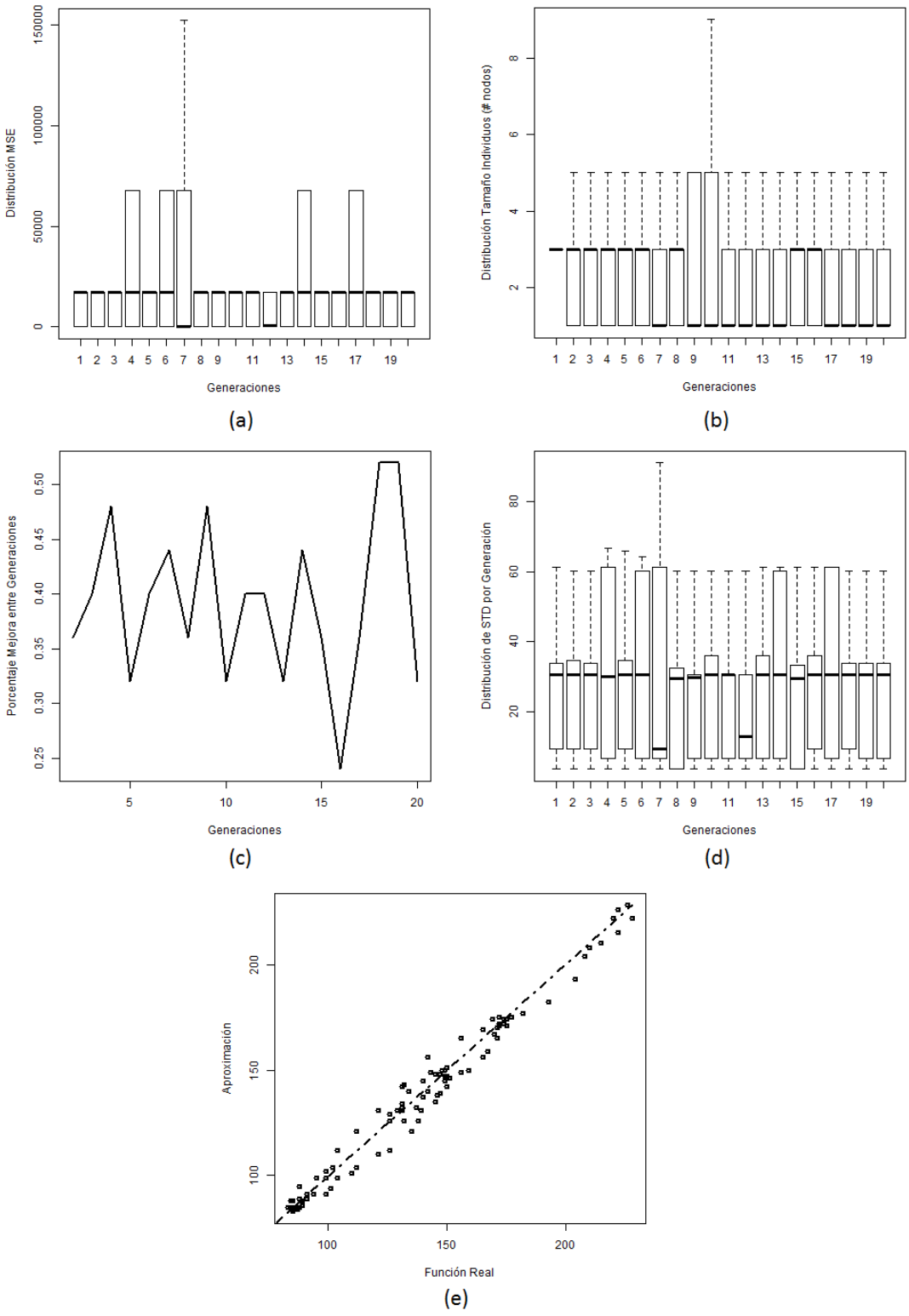


Figura 3-14: Criterios de evaluación del modelo de predicción generado por GP (tradicional) para la serie INTERNET: a) Gráfico de boxplot de la aptitud de la población por generación; b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación; c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación; d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación; e) Gráfico de la relación de aproximación vs valor real.

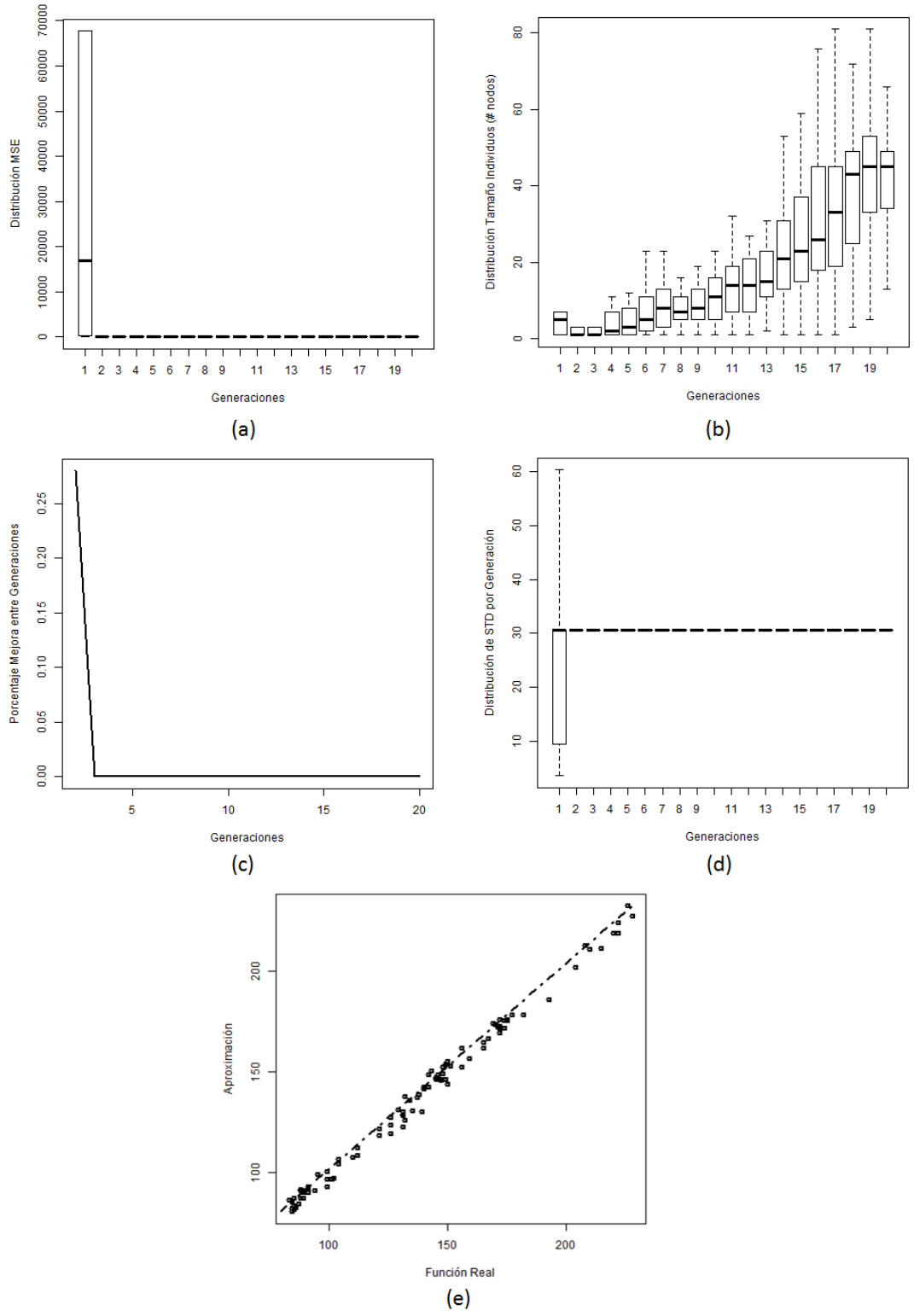
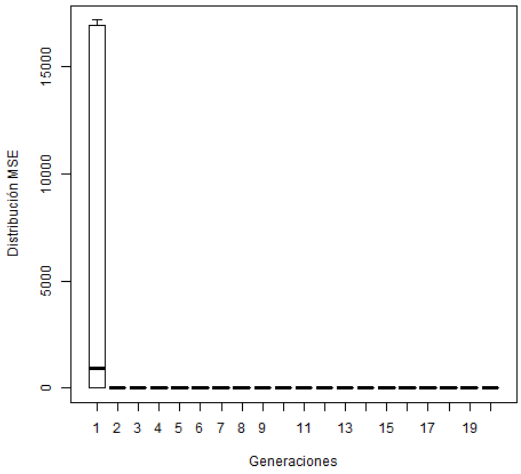
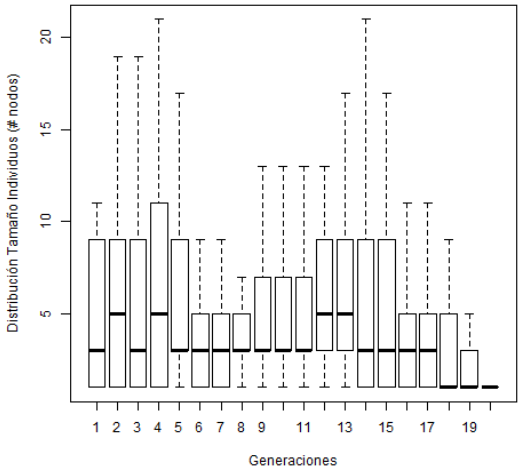


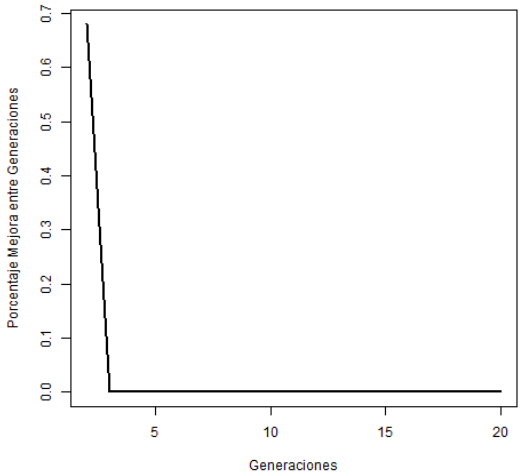
Figura 3-15: Criterios de evaluación del modelo de predicción generado por gpModel para la serie INTERNET: a) Gráfico de boxplot de la aptitud de la población por generación; b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación; c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación; d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación; e) Gráfico de la relación de aproximación vs valor real.


Distribución MSE
0
5000
10000
15000
1 2 3 4 5 6 7 8 9 11 13 15 17 19
Generaciones

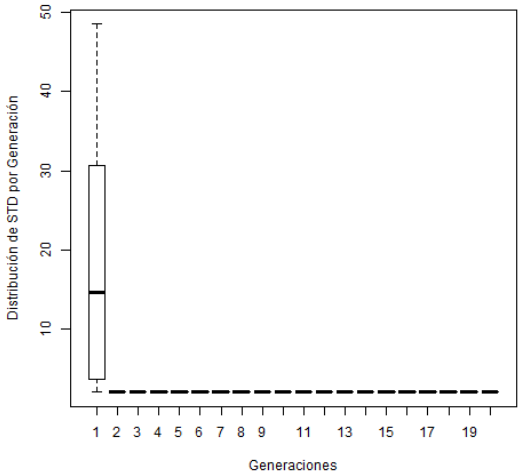
(a)


Distribución Tamaño Individuos (# nodos)
5
10
15
20
1 2 3 4 5 6 7 8 9 11 13 15 17 19
Generaciones

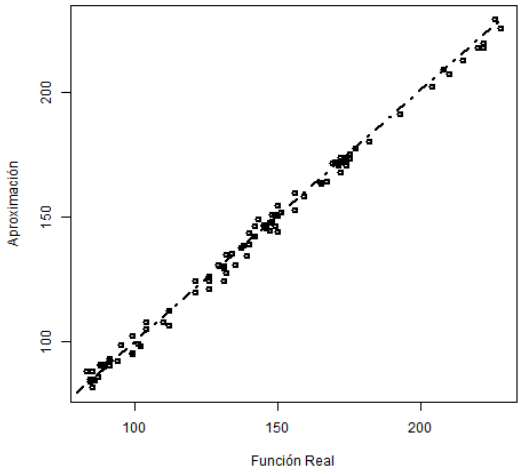
(b)


Porcentaje Mejora entre Generaciones
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
5 10 15 20
Generaciones

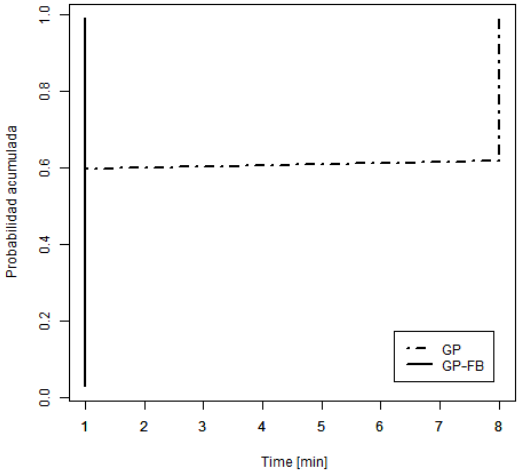
(c)


Distribución de STD por Generación
10
20
30
40
50
1 2 3 4 5 6 7 8 9 11 13 15 17 19
Generaciones

(d)


Aproximación
100
150
200
100 150 200
Función Real

(e)


Probabilidad acumulada
0.0
0.2
0.4
0.6
0.8
1.0
1 2 3 4 5 6 7 8
Time [min]
GP
GP-FB

(f)

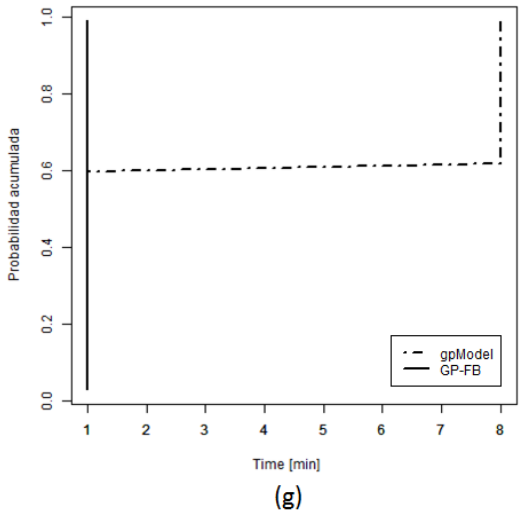


Figura 3-16: Criterios de evaluación del modelo de predicción generado por GP-FB para la serie INTERNET: a) Gráfico de boxplot de la aptitud de la población por generación; b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación; c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación; d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación; e) Gráfico de la relación de aproximación vs valor real; f) Gráfica Time-to-Target (tttplot), entre los modelos GP (tradicional) y GP-FB; g)) Gráfica Time-to-Target (tttplot), entre los modelos gpTool y GP-FB.

Si bien los cambios propuestos por Martínez [12] generan mejoras en la modelación con GP, aún se muestran falencias en los criterios de aceptación de los modelos resultantes. Lo anterior se evidencia en la Figura 3-15, en la que se muestran los criterios de evaluación del modelo de predicción generado por gpModel para la serie INTERNET; en a) se muestra que el comienza con un SSE = 18.000 en la primera generación, disminuyendo en las siguientes asociado al uso de los bloques funcionales (gráfico de boxplot de la aptitud de la población por generación); lo que conlleva a una disminución de la desviación estándar centrada en 30, como se muestran en d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación; Sin embargo, en b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación, muestra un crecimiento exponencial en el número de nodos (con un máximo de 80 nodos) debido principalmente a la metodología de cruce con suma sin poda. Por otra parte, en c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación, se muestra un decrecimiento a partir de la tercera generación (desde el 28%), en línea con los demás graficos, en el que se muestra que una vez ha alcanzado un individuo con menor medida de error, las modificaciones realizadas por los operadores genéticos en la población no impactan realmente en el resultado (generación de bloat), y cuyos resultados se reglejan en e) Gráfico de la relación de aproximación vs valor real, en cuyo caso es mejor que GP, pero que sigue alejado del valor objetivo correspondiente a la recta con ángulo de 45 grados.

Por otra parte, en la Figura 3-16 se muestran los resultados del algoritmo propuesto con las modificaciones planteadas en el Capítulo 2, en el que claramente se aprecia una disminución en la medida de error desde la primera generación (centrado en SSE = 1.000 y un máximo de SSE = 18.000), cuyas mejoras son replicadas a los demás individuos de la población, llegando a un nivel general en la tercera generación (SSE = 296, equivalente a un MSE = 3,7), como se puede ver en a) Gráfico de boxplot de la aptitud de la población por generación. Al igual que gpModel, GP-FB tendría un crecimiento desmedido en los individuos, pero es efectivamente controlado por medio del operador de simplificación (Subsección 2.2.10) por lo que en b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación, se muestra un crecimiento lineal con un promedio de 3 nodos y un máximo de 21 nodos. Debido principalmente a la rápida convergencia

de los individuos, c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación, tiende a disminuir con tendencia a cero desde un 68% inicial de mejora entre las dos primeras generaciones. Todo lo anterior influye positivamente en d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación, en el que tienen tendencia decreciente (mucho menor a GP y gpModel) y media cercana a cero; y e) Gráfico de la relación de aproximación vs valor real, en el que se muestra de nuevo (al igual que la Figura 3-13), la capacidad del algoritmo de encerrar el comportamiento de la serie (centrada sobre la recta con ángulo de 45 grados). Igualmente, en el análisis de convergencia en f) Gráfica Time-to-Target (tttplot), entre los modelos GP (tradicional) y GP-FB, y g) Gráfica Time-to-Target (tttplot), entre los modelos gpModel y GP-FB, se concluye que GP-FB tiene una alta probabilidad de converger a un valor de aptitud (o error) dado en muy poco tiempo, para el caso particular de la serie INTERNET en el primer minuto, en contraposición a GP que lo hace en 8 minutos y gpModel también en 8 minutos.

3.3.5 Serie SUNSPOT

La serie SUNSPOT describe la cantidad anual de manchas solares observadas en el sol a partir del año 1700. Estos datos presentan alta no linealidad y han sido usados para medir la efectividad de aproximación de estadísticos no lineales. En la literatura se suele modelar la serie original sin ningún tipo de transformación, tomando los primeros 221 datos para el entrenamiento y los 35 restantes para la predicción.

Cottrell et al. [127] analizan la serie a partir de redes neuronales (NN, por su sigla en inglés), tomando como periodo de entrenamiento los años entre 1700 y 1921 (221 datos) y los años 1921 al 1956 (35 puntos) de pronóstico. Para lo anterior, plantean dos esquemas de NN en los cuales, el primer modelo utilizaba una capa oculta de entre 3 y 5 neuronas, 4 rezagos de entrada y una neurona en la capa de salida; mientras el segundo consideraba 11 rezagos a la entrada, una capa oculta con dos neuronas y una capa de salida con solo una neurona; siendo esta configuración la de mejor resultado en términos de MSE.

Por otro lado, Zhang [125] utilizó la serie como referencia para evaluar la efectividad del modelo hibrido propuesto comparado con los modelos de redes neuronales y ARIMA. Al igual que Cotter et al. [127], los primeros 221 datos son utilizados como entrenamiento, pero extiende la cantidad de puntos de pronóstico a 67, realizando comparativos con los 35 primeros para pronóstico con los resultados registrados en la literatura. Para realizar los comparativos se apoya en las medidas de error de MSE y MAD cuyos resultados están consignados en la Tabla 3-5.

Tabla 3-5: Resultados de entrenamiento y pronóstico entre modelos de predicción para la serie SUNSPOT.

Modelo	Rezagos	Entrenamiento MSE (MAD)	Predicción MSE (MAD)
GP-FB*	**1 – 11**	**37 (3,10)**	**75 (3,90)**
gpModel**	**1 – 11**	**47 (3,90)**	**85 (4,70)**
ARIMA[2]	N/A	N/A	217 (11,30)
GP[2]	1-11	853 (14,83)	2.184 (31,46)
ANN[2]	N/A	N/A	205 (10,20)
Hibrido[2]	N/A	N/A	187 (10,80)
DAN2-1[2]	1,3,4,9,10,11	78 (7,00)	145 (9,70)
DAN2-2[2]	1,2,9,11	95 (7,40)	146 (9,60)
DAN2-3[2]	1,2,3,9,10,11	120 (8,40)	186 (9,90)

* Generación propia; Test Wilcox p-value = 0.002 con respecto a gpModel.

** Martínez [12]

[2]Ghiassi et al. [16]

Adicionalmente, Ghiassi et al.[16] analizaron la serie en virtud de los modelos DAN2, para lo cual tomaron los primeros 221 puntos de entrenamiento y los restantes 67 de validación, siendo los últimos 35 más utilizados para la comparación con otros métodos. En este trabajo desarrollaron 3 modelos en los cuales fueron tomados distintos rezagos entre el primero y el décimo primero, siendo el primero el mejor en términos de las medidas de MSE y MAD.

En este trabajo se aplicó tanto el modelo de GP original [3] y el algoritmo modificado (de acuerdo con la metodología propuesta en el Capítulo 2) GP-FB, en el cual los primeros 221 datos son utilizados para la estimación del modelo y los 35 restantes para su pronóstico. Para la ejecución del algoritmo modificado, fueron utilizados 150 individuos de población inicial (3 niveles de profundidad y 7 terminales a lo máximo), un número máximo de 15 generaciones, función de error SSE, algoritmo de optimización OPTIM [109] y de profundización RGNOUD [120] (ver Subsección 3.2).

En la Figura 3-17 se puede apreciar tanto la serie original SUNSPOT y el mejor modelo GP-FB (de acuerdo con la corrida realizada con los parámetros antes mencionados), para los datos de entrenamiento (lado izquierdo de la línea vertical) y de pronóstico (línea discontinua y círculos vacíos), la cual muestra que el modelo realmente encierra el comportamiento de los datos y no solo los memoriza (problema común de falta de generalización en la predicción de series de tiempo); esto también puede ser evidenciado en la similitud entre los errores de entrenamiento y predicción en SSE y una menor volatilidad MAD (Tabla 3-5).

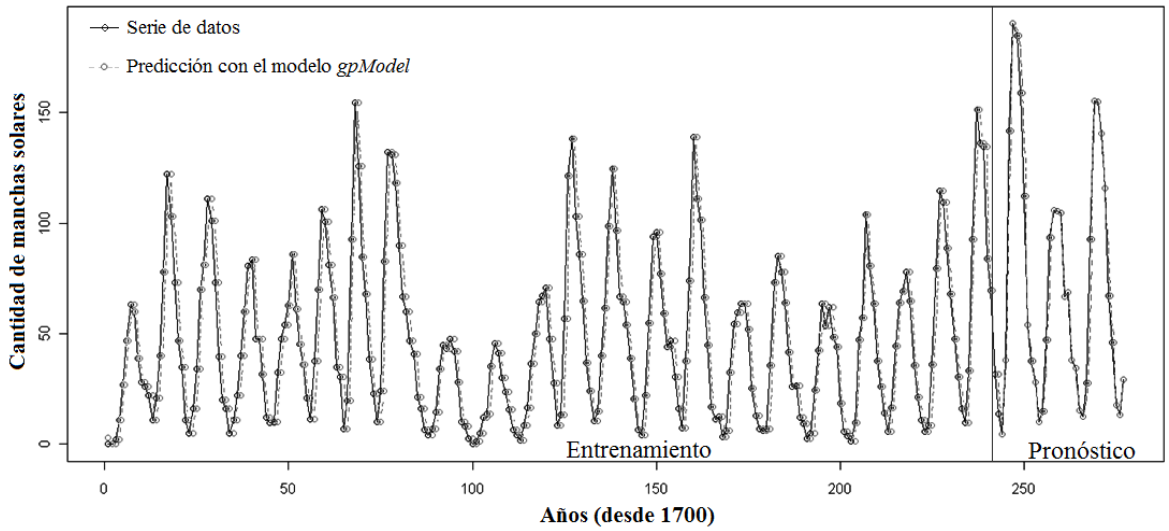


Figura 3-17: Resultados de entrenamiento y pronóstico del modelo de predicción para la serie SUNSPOT por medio de la metodología propuesta.

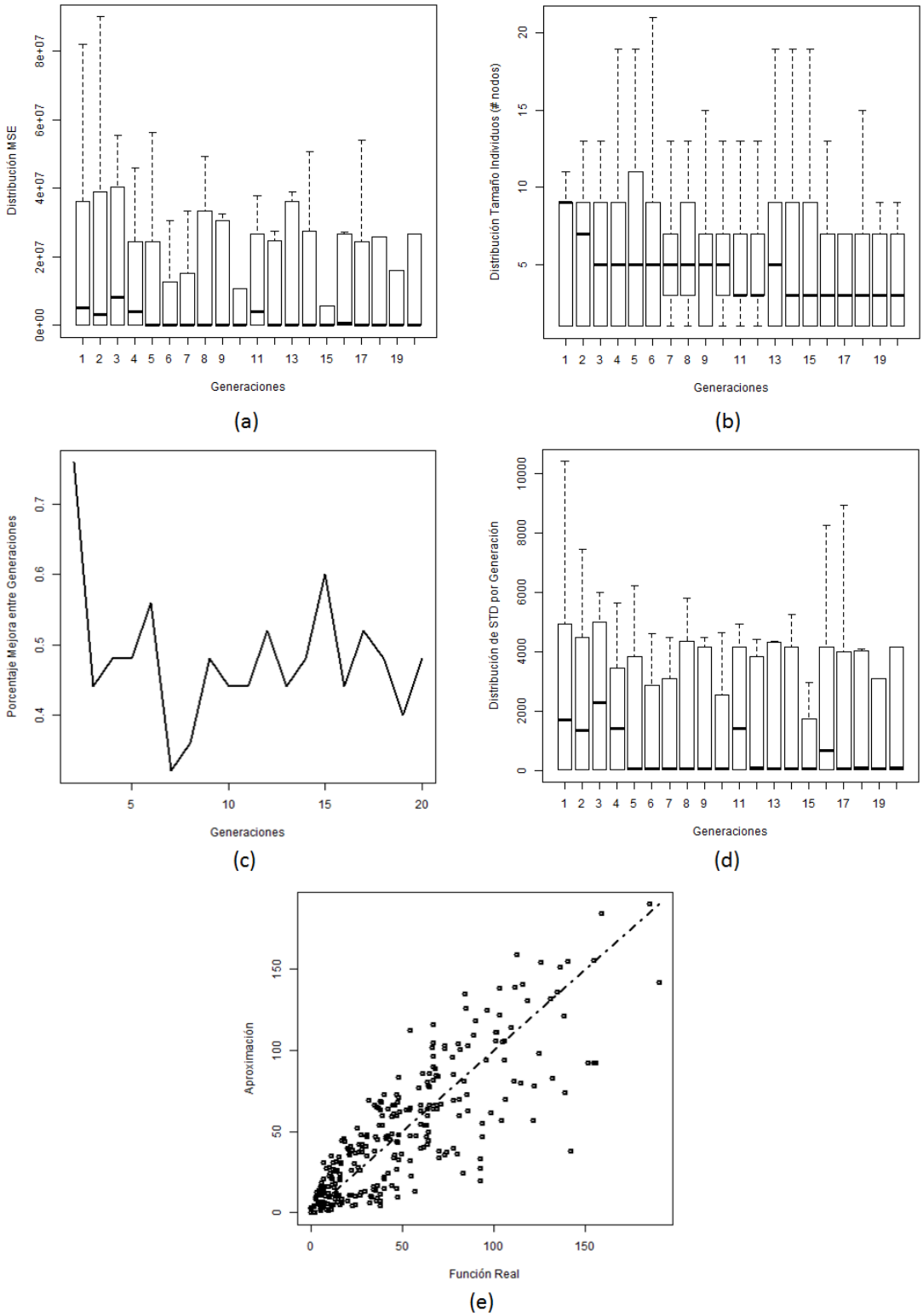


Figura 3-18: Criterios de evaluación del modelo de predicción generado por GP (tradicional) para la serie SUNSPOT: a) Gráfico de boxplot de la aptitud de la población por generación; b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación; c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación; d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación; e) Gráfico de la relación de aproximación vs valor real.

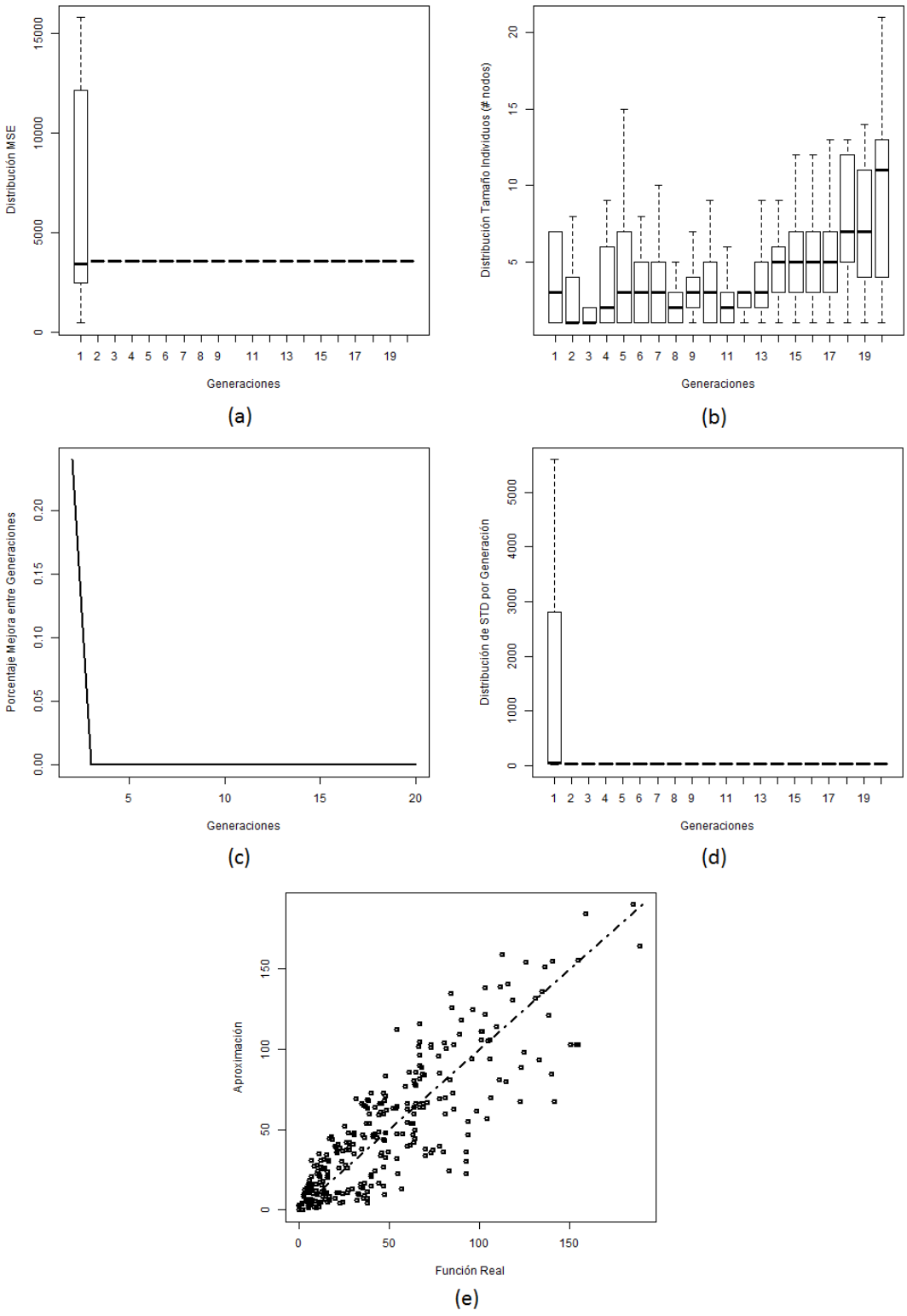
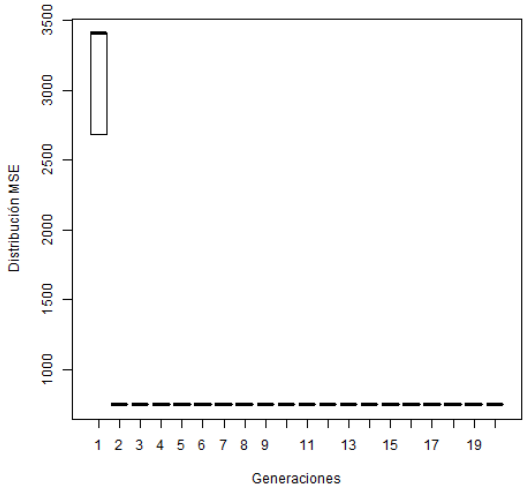
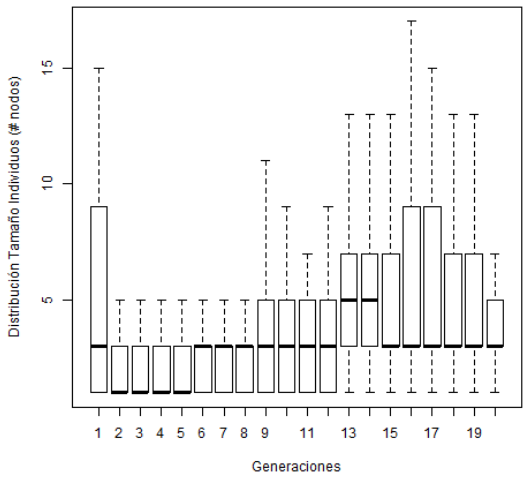


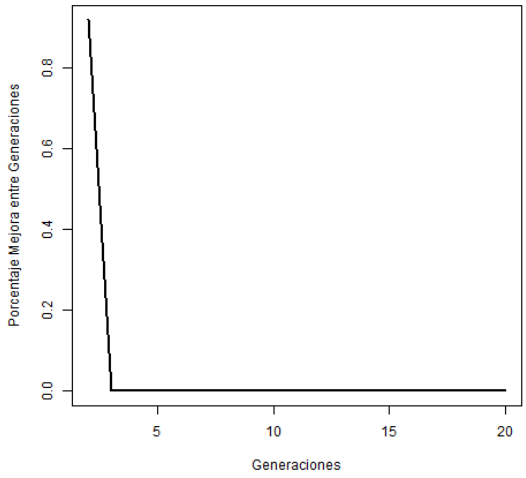
Figura 3-19: Criterios de evaluación del modelo de predicción generado por gpModel para la serie SUNSPOT: a) Gráfico de boxplot de la aptitud de la población por generación; b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación; c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación; d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación; e) Gráfico de la relación de aproximación vs valor real.


Distribución MSE
Generaciones

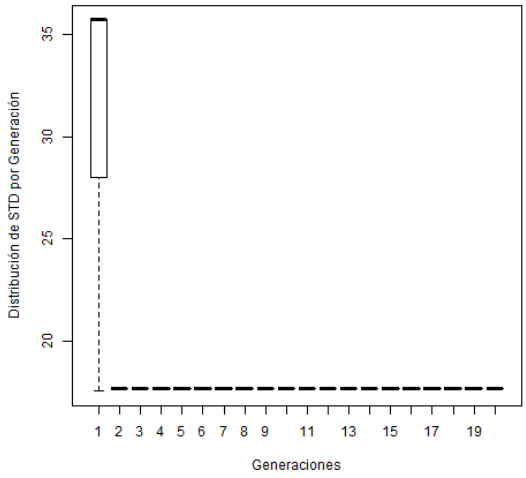
(a)


Distribución Tamaño Individuos (# nodos)
Generaciones

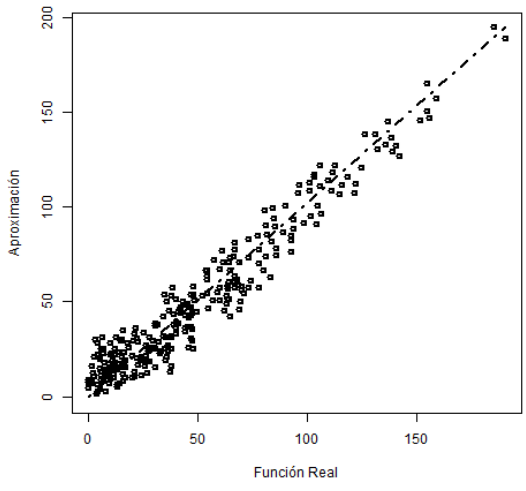
(b)


Porcentaje Mejora entre Generaciones
Generaciones

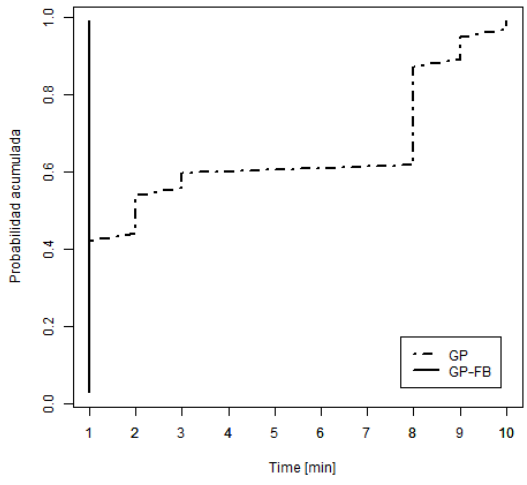
(c)


Distribución de STD por Generación
Generaciones

(d)


Aproximación
Función Real

(e)


Probabilidad acumulada
GP
GP-FB
Time [min]

(f)

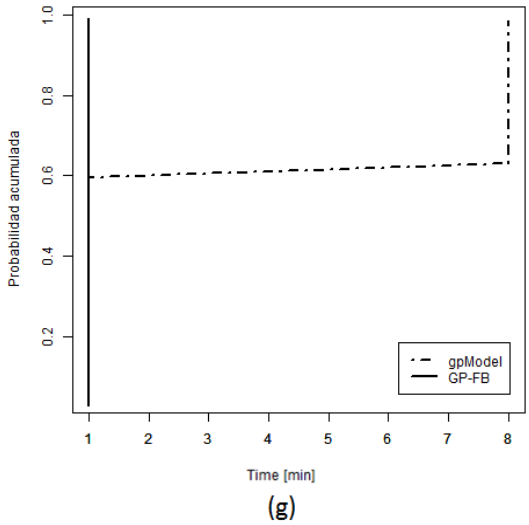


Figura 3-20: Criterios de evaluación del modelo de predicción generado por GP-FB para la serie SUNSPOT: a) Gráfico de boxplot de la aptitud de la población por generación; b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación; c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación; d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación; e) Gráfico de la relación de aproximación vs valor real; f) Gráfica Time-to-Target (tttplot), entre los modelos GP (tradicional) y GP-FB; g)) Gráfica Time-to-Target (tttplot), entre los modelos gpTool y GP-FB.

Los resultados, tanto de entrenamiento como de pronóstico que se muestran en la Tabla 3-5, en la cual se comparan resultados obtenidos en la literatura con modelos ARIMA, NN, DAN2 e Hibrido, adicional a los modelos de GP original identificado por GP, el modelo modificado propuesto en la tesis de maestría [12] identificado por gpModel, y el modelo propuesto GP-FB; en el cual se puede apreciar que el modelo GP-FB presenta valores de MSE de entrenamiento inferiores al de DAN2 en todas sus variantes (52,6%) además de un menor valor con respecto al algoritmo original GP (95,7%); en los demás modelos no es posible la comparación dada la falta de información en la literatura. Por otro lado, los resultados reportados en las fases de pronóstico y predicción son mejores en GP-FB con respecto a los demás registrados (en términos del valor del MSE), con mejoras que van desde el 59,9% con respecto al modelo Hibrido (ARIMA-ANN con mejora del 63,9% en el MAD) al 65,4% con respecto modelo ARIMA (mejora del 65,5% en el MAD), pasando por el 63,4% con respecto a registrado por NN (mejora del 61,8% en el MAD). Cabe resaltar que en comparación con los resultados obtenidos en la predicción con respecto a los modelos DAN2 registra una mejoría del 48,3% en la medida de MSE y del 59,8 % en el MAD, lo que sugiere que son superiores en capacidad de predicción de la serie. Adicionalmente, comparando el modelo obtenido GP-FB con el algoritmo original de GP también reporta una mejora del 95,7% en el error de aproximación y del 96,6% en el de predicción; y con respecto al modelo gpModel reporta una mejora del 21,3% en el error de aproximación y del 11,8% en el de predicción.

En la Figura 3-18 se muestran los criterios de evaluación del modelo de predicción generado por GP (tradicional) para la serie SUNSPOT: a) Gráfico de boxplot de la aptitud de la población por generación; b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación; c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación; d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación; e) Gráfico de la relación de aproximación vs valor real. En los cuales se puede apreciar que los modelos generados por GP exhiben una alta distribución del error (a) llegando inclusive a SSE = 8e+7 y repercutiendo en los valores de la desviación estándar (d) cercano al 2.000, aunque mantiene estable el tamaño de los individuos (b) entre 1 y 20 nodos; debido principalmente a la aplicación del operador de cruce tradicional. Sin una estabilización clara de las mejoras de la proporción de

individuos mejores (menor medida de error – mayor valor de Aptitud, oscilando entre el 30% y el 78%) entre generaciones (c) e influyendo claramente en su aproximación (e) el cual está claramente muy disperso del valor esperado (línea recta con ángulo de 45 grados).

Si bien los cambios propuestos en la tesis de maestría [12] generan mejoras en la modelación con GP, aún se muestran falencias en los criterios de aceptación de los modelos resultantes. Lo anterior se evidencia en la Figura 3-19, en la que se muestran los criterios de evaluación del modelo de predicción generado por gpModel para la serie SUNSPOT; en a) se muestra cecntrado en SSE = 4.000 desde la primera generación (gráfico de boxplot de la aptitud de la población por generación); lo que conlleva a una disminución de la desviación estándar centrada en 90, como se muestran en d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación; Sin embargo, en b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación, muestra un crecimiento (tendiendo a exponencial) en el número de nodos debido principalmente a la metodología de cruce con suma sin poda (con un máximo de 20 nodos). Por otra parte, en c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación, se muestra un decrecimiento a partir de la tercera generación (partiendo del 25%), en línea con los demás graficos, en el que se muestra que una vez ha alcanzado un individuo con menor medida de error, las modificaciones realizadas por los operadores genéticos en la población no impactan realmente en el resultado (generación de bloat), y cuyos resultados se reglejan en e) Gráfico de la relación de aproximación vs valor real, en cuyo caso es mejor que GP, pero que sigue alejado del valor objetivo correspondiente a la línea de 45 grados.

Por otra parte, en la Figura 3-20 se muestran los resultados del algoritmo propuesto con las modificaciones planteadas en el Capítulo 2, en el que claramente se aprecia una disminución en la medida de error desde la primera generación (centrado en SSE = 3.400), cuyas mejoras son replicadas a los demás individuos de la población, llegando a un nivel general en la segunda generación (MSE = 37), como se puede ver en a) Gráfico de boxplot de la aptitud de la población por generación. Al igual que gpModel, GP-FB tendría un crecimiento desmedido en los individuos, pero es efectivamente controlado por medio del operador de simplificación (Subsección 2.2.10) por lo que en b) Gráfico de boxplot del tamaño de la población (medio en nodos) por generación, se muestra un crecimiento lineal con un promedio de 3 nodos y un máximo de 16 nodos. Debido principalmente a la rápida convergencia de los individuos, c) Gráfico de la proporción de la población con mejora en el valor de aptitud por generación, tiende a disminuir con tendencia a cero desde un 87% inicial de mejora entre las dos primeras generaciones. Todo lo anterior influye positivamente en d) Gráfico de boxplot de la desviación estándar del valor de aptitud de la población por generación, en el que tienen tendencia decreciente (mucho menor a GP y gpModel) y media cercana a cero; y e) Gráfico de la relación de aproximación vs valor real, en el que se muestra de nuevo (al igual que la Figura 3-17), la capacidad del algoritmo de encerrar el comportamiento de la serie (centrada sobre la recta con ángulo de 45 grados). Igualmente, en el análisis de convergencia en f) Gráfica Time-to-Target (tttplot), entre los modelos GP (tradicional) y GP-FB, y g) Gráfica Time-to-Target (tttplot), entre los modelos gpModel y GP-FB, se concluye que GP-FB tiene una alta probabilidad de converger a un valor de aptitud (o error) dado en muy poco tiempo, para el caso particular de la serie SUNSPOT en el primer minuto, en contraposición a GP que lo hace en 10 minutos y gpModel en 8 minutos.

3.4 Conclusiones

Debido a que las series de datos seleccionadas han sido ampliamente utilizadas en la comparación de modelos de predicción de series de tiempo dadas sus distintas propiedades estadísticas, y que el algoritmo tradicional de GP y el algoritmo propuesto en este trabajo (GP-FB) fueron ejecutados bajos los mismos criterios de inicialización y funciones de error seleccionadas; los resultados obtenidos son válidos en la comparación de la capacidad de predicción entre los distintos algoritmos, resaltándose que la versión GP-FB obtuvo una menor

medida de error en los datos de entrenamiento y de validación/predicción para todas las series benchmark analizadas.

El uso del algoritmo modificado de GP (GP-FB) sugiere una mejor aproximación y predicción a la serie de datos original en virtud de las medidas de error MSE, SSE y MAD, lo que permite una modelación más cercana a la serie.

En este capítulo se mostró la capacidad de generación de modelos de predicción de las series de datos, enmarcadas en los resultados de pronóstico de cinco de los principales benchmark de la literatura de análisis y predicción de series de tiempo, superando en cada una de ellas los valores registrados en la literatura de predicción de series de tiempo. Adicional a la capacidad de convergencia en poco tiempo, superando en todos los casos al algoritmo original de GP y el propuesto en la tesis de maestría, considerando además que todos fueron implementados con las mejoras de rendimiento propuesto en la sección 2.2.14.

Es de tener en cuenta que en el algoritmo propuesto GP-FB es posible la configuración de varios parámetros de aplicación de los operadores genéticos, la selección del conjunto de terminales (T) y operadores (Z), los cuales deben ser definidos a partir del análisis de las propiedades estadísticas de la serie y el conocimiento acerca del fenómeno por parte del investigador, lo cual conllevará a una mejora en la capacidad de predicción del modelo resultante. Se resalta que la ejecución del algoritmo de GP-FB en cada una de las series de tiempo de este capítulo, fue realizado de acuerdo con los parámetros descritos en la Subsección 3.2, sin cambio entre una u otra serie temporal (exceptuando el número de rezagos a ser considerados).

Por último, debido a que los terminales del algoritmo GP-FB están compuestos exclusivamente de bloques funcionales, y éstos a su vez fueron definidos a partir de las componentes de los principales modelos de predicción de series de tiempo presentes en la literatura, las soluciones/individuos resultantes de GP-FB son hibridaciones de ellos y por ende es posible una interpretación más cercana a los modelos existentes —de predicción de series de tiempo— y a las características propias de la serie de tiempo.

4. Conclusiones y trabajo futuro

4.1 Respuesta a las preguntas de investigación

En este trabajo fue analizado el algoritmo original de GP, sus principales mejoras, cambios estructurales e hibridaciones presentes en la literatura actual; identificando además, las principales falencias del algoritmo en la predicción de series de tiempo, planteando para ellas, una serie de modificaciones que implicaron cambios en la estructura de los individuos y pasos del algoritmo, en pro de una mejor predicción, un proceso de búsqueda más focalizada y unas ecuaciones resultantes más simples.

Entre las modificaciones presentadas, se resalta la inclusión de los BF como constituyentes de los principales modelos de predicción de series de tiempo actuales en la literatura, dichos componentes permitieron reescribir las ecuaciones de los modelos de predicción de una manera más clara, concisa y de fácil manejo en la hibridación entre los mismos. Además, el cambio en la probabilidad de selección de los bloques funcionales durante el proceso de búsqueda, la redefinición de los operadores genéticos de cruce y mutación, la aplicación de nuevos operadores genéticos, la simplificación de los árboles y la inclusión de la intensificación como paso adicional de ajuste del modelo, permitieron generar modelos más aproximados a la realidad del fenómeno descrito por la serie de tiempo (error de predicción menor), con menor tamaño y claramente identificables de acuerdo con los modelos presentes en la literatura de predicción de series temporales; inclusión de conocimiento experto y focalización del espacio de búsqueda.

Por otra parte, fue mostrado el método de inclusión de los BF en el algoritmo original de GP, y modificaciones propuestas en la literatura, a partir de su uso como componentes fundamentales en los terminales de los individuos, lo que impactó positivamente en la forma y estructura de las soluciones encontradas con esta técnica, adicional a la inclusión de los distintos modelos de predicción a partir de la definición de los respectivos BF correspondientes.

Las modificaciones al algoritmo original de GP fueron validados contra modelos lineales, no lineales estándar y series benchmark de la literatura, verificando su efectividad en la deducción de la estructura y aproximación de los parámetros que mejor describe las observaciones, teniendo en cuenta que dicho proceso de aproximación depende en gran medida del algoritmo de optimización utilizado.

De acuerdo con los resultados obtenidos y la estructura del algoritmo planteada y desarrollada, es posible la incorporación de conocimiento a priori representado en la generación de nuevos bloques funcionales, selección de los parámetros de ejecución, generación de la población inicial, incorporación de nuevos criterios de parada, estructura de generación de los individuos en la población y la especificación de los parámetros generales y los respectivos coeficientes de los individuos a ser utilizados durante la ejecución del mismo; permitiendo una gran mejora en la aproximación (disminución del error de aproximación) —al poder ser evaluados nuevos tipos de modelos y relaciones—, y en el tiempo de recorrido —al limitar el espacio de búsqueda—.

A continuación se presentan las respuestas obtenidas en estas tesis a las preguntas de investigación formuladas en el Capítulo 1, correspondientes a cada una de las siguientes subsecciones.

4.1.1 ¿Es posible ampliar el conjunto de terminales a ser utilizados en el algoritmo de programación genética introduciendo información relevante que realmente influya positivamente en el error de aproximación del modelo resultante?

En este trabajo fue demostrado que el uso de bloques funcionales como constituyentes fundamentales y exclusivos de los terminales permite la inclusión de un número mayor de los terminales (no solo los rezagos) asociados a modelos de predicción de series de tiempo ampliamente utilizados en la literatura, y por ende, pueden disminuir el error de predicción de la solución encontrada al aplicar el algoritmo de GP, a un costo computacional aceptable.

Adicionalmente, se implementó un paso de pre procesamiento de la serie de tiempo, que permite identificar sus propiedades estadísticas y ajustar la probabilidad de selección de los bloques funcionales, los parámetros de aplicación de los operadores genéticos, y la focalización del espacio de búsqueda en zonas de interés. Cada uno de los bloques funcionales fueron agrupados en una de las componentes de la serie de tiempo, por lo que es posible aumentar la selección de algunos BF dependiendo de las características de la serie temporal y por ende incluir todos aquellos que son de relevancia para la misma.

4.1.2 ¿Cómo incluir conocimiento experto para focalizar la búsqueda sobre los modelos estructuralmente más prometedores?

Dado que los bloques funcionales son funciones que toman como entradas todos los rezagos definidos para el modelo de predicción; la optimización de los parámetros de cada uno de los terminales determina la importancia del rezago en el individuo, permitiendo así la inclusión de aquellos que pueden ser de relevancia para la predicción de la serie de tiempo.

Entre los distintos cambios propuestos, se incluye la modificación de los operadores de cruce y mutación para la focalización de las estructuras en zonas de interés por medio de un modelo de agregación, y la actualización dinámica de las probabilidades de aplicación de cada uno de ellos durante el proceso de búsqueda a partir de la información de las poblaciones anteriores. Se propuso el uso de medidas de aptitud basadas en criterios de información, que permite focalizar los individuos en estructuras de menor error de predicción ponderando el tamaño del mismo; además, al final del algoritmo, fue propuesto un paso adicional de intensificación, el cual pretende optimizar los parámetros en la estructura resultante (modelo) por medio de otras técnicas adicionales no basadas en gradiente.

4.1.3 ¿Es posible reducir las operaciones redundantes de los individuos sin afectar el proceso de exploración del algoritmo de programación genética?

Debido a que el uso de bloques funcionales reemplaza los distintos terminales originales (en los que se tomaban solo los rezagos), por una función que toma todos los rezagos, las operaciones redundantes son mínimas y fácilmente asociables a modelos de predicción de series de tiempo presentes en la literatura.

Por otra parte, se realizó un análisis de las medidas de error basadas en criterios de información adaptadas al uso de programación genética, de las cuales se seleccinó el criterio Akaike como medida de error, en la que se pondera la capacidad de predicción (error) y el tamaño del individuo de GP, con el fin de disminuir el bloat. Adicionalmente, se generaron estrategias de poda del árbol basado principalmente en la interpretación de las propiedades de simplificación matemática aplicada a los individuos de GP modificados con bloques funcionales.

4.1.4 ¿Es posible introducir conocimiento experto durante la generación de los individuos del algoritmo de programación genética que permita su consistencia lógico-matemática?

Debido a que fueron redefinidos los terminales como bloques funcionales y estos corresponden a funciones tomadas de los principales modelos de predicción de series de tiempo, los modelos/individuos resultantes

corresponden a la hibridación de dichos modelos de predicción a partir del uso de operadores aritméticos básicos $(+, -, /, *)$, lo que permite que posea una consistencia matemática con respecto a los modelos de predicción de series de tiempo presentes en la literatura y una equivalencia del modelo resultante más clara con dichos modelos de predicción de series de tiempo.

Cabe resaltar que cada uno de los bloques funcionales pertenece a una de las componentes de la serie de tiempo, por lo que se modeló el conjunto de terminales por medio de cuatro subconjuntos de bloques funcionales (Ciclo, Tendencia, Estacionalidad, Error), y se realizó una combinación lineal de los mismos, en la cual se analiza las características de la serie temporal para aumentar la probabilidad de uso de los subconjuntos más afines. Por otra parte, durante el proceso de búsqueda, se valida la estructura de los individuos, respetando las reglas de combinación entre los distintos subconjuntos de bloques funcionales, por lo que se mantiene la consistencia lógico-matemática durante todo el algoritmo.

Lo anterior, muestra que fue posible responder a cada una de las preguntas durante el trabajo de investigación, proponiendo un nuevo algoritmo de programación genética (GP-FB) en el cual se reestructura cada uno de los componentes del mismo, en pro de una búsqueda más focalizada y de mejor comportamiento en la predicción de la serie real (menor error de predicción).

4.2 Alcance de los objetivos propuestos en la tesis

Respecto a los objetivos específicos planteados en la investigación se tienen:

4.2.1 Proponer un nuevo mecanismo de generación de individuos de programación genética por medio de agrupaciones de bloques funcionales, con el fin de garantizar individuos lógicos de manera matemática, y su posterior interpretación a la luz de las variables propias de modelación

En este trabajo se redefinió la estructura de los individuos del algoritmo original de GP incorporando los bloques funcionales como componentes fundamentales y exclusivos de los terminales, cada uno de los bloques corresponde a una función proveniente de un modelo de predicción de series de tiempo presente en la literatura (Capítulo 2 y Anexo A), el cual permite una estructura de las soluciones más simple, fácil de interpretar y que encierra de mejor manera (menor error de predicción) el comportamiento de la serie de datos. Verificando que es posible la incorporación de las funciones más ampliamente utilizadas en la predicción de series de tiempo a partir del uso de los BF como componentes de las ecuaciones.

Adicionalmente, fueron agrupados los bloques funcionales de acuerdo a la componente de la serie de tiempo (Ciclo, Tendencia, Estacionalidad, Error), por lo que los individuos fueron reescritos como una combinación lineal de cada una de las componentes, las cuales a su vez son una combinación lineal de los bloques funcionales, permitiendo mantener la consistencia matemática y una relación más directa con los modelos de predicción de series temporales, lo cual sugiere una interpretación más clara del aporte de cada una de las variables/rezagos a la modelación.

Lo anterior deja la puerta abierta a nuevos desarrollos que no solo involucren variables endógenas sino la incorporación de valores exógenos de validez para la predicción de la serie, al igual que la definición de nuevos tipos de comportamientos como el quiebre de políticas (series por tramos), que si bien no fueron objetos de esta investigación es posible su incorporación por medio de operadores lógicos y condiciones de evaluación del algoritmo.

4.2.2 Modificar el proceso de generación de los individuos de programación genética para garantizar la inclusión de cada uno de los rezagos de interés, con el fin de disminuir la influencia de los terminales en la selección de la estructura funcional de los individuos

En el Capítulo 2 de este trabajo, se redefinió la estructura de los individuos del algoritmo original de GP incorporando única y exclusivamente bloques funcionales como componentes de los terminales, se agruparon de acuerdo a las componentes de las series temporales (Ciclo, Tendencia, Estacionalidad, Error); cada uno de

los bloques corresponde a una función proveniente de un modelo de predicción de series de tiempo presente en la literatura (Capítulo 2 y Anexo A), el cual permite una estructura de las soluciones más simple, fácil de interpretar y que encierra de mejor manera (menor error de predicción) el comportamiento de la serie de datos.

Cabe resaltar que debido a que los individuos son generados por los operadores y terminales, y las terminales son constituidas únicamente por bloques funcionales, y estos últimos corresponden a una expresión matemática completa que incluye tanto todos los rezagos a ser utilizados, como los parámetros/coeficientes y constantes externas, se puede garantizar el uso de todos los rezagos de interés en cada uno de los individuos durante todo el proceso de búsqueda del algoritmo de GP; y dado que se utiliza un algoritmo que cumple las condiciones de similitud de valor de aptitud a estructuras funcionales similares (Subsección 2.2.5), se puede afirmar que los mejores individuos (en términos del valor de la función de aptitud seleccionada) corresponden a aquellos que poseen una estructura funcional más cercana con la función generadora de la serie de tiempo.

Por otra parte, dado que cada uno de los bloques funcionales tiene inmerso los parámetros propios de la expresión matemática equivalente, no es necesario el uso de constantes y por ende, una disminución clara del tamaño de los individuos.

4.2.3 Modificar los operadores genéticos para permitir la focalización del espacio de búsqueda del algoritmo de programación genética en zonas prometedoras de búsqueda

En este trabajo se modificaron los operadores genéticos de cruce y mutación del algoritmo original de GP, permitiendo el crecimiento de los individuos (adición de nuevos operadores y terminales) alrededor de una zona de búsqueda de interés, logrando así la focalización de esfuerzos en regionales específicas, validando los resultados de la metodología propuesta contra series de tiempo con ecuación de generación conocida, y verificando que permite una convergencia más rápida y una solución más cercana a la ecuación original.

Para el operador genético de cruce se optó por un equilibrio entre la definición tradicional de GP y la recombinación de los individuos padre en un único individuo hijo (Subsección 2.2.8), con el fin de lograr un equilibrio entre la intensificación y la diversificación. La recombinación de los individuos permite redirigir la población hacia zonas en las cuales el valor de aptitud de los individuos hijo son superiores a los de los individuos padre, en contraposición a la versión original de GP que tan solo se distribuia de forma aleatoria sin ninguna garantía de mejora.

Para el operador genético de mutación se optó por el cambio en la dirección de mejora del individuo (Subsección 2.2.7) en la que se generan pequeños cambios en los individuos cambiando un único nodo que corresponde al de mayor destrucción de valor de aptitud al interior del individuo, garantizándose por ende una mejora del mismo a la siguiente generación sin un aumento en el tamaño.

Adicional a lo anterior, se realizó la respectiva redefinición de los demás operadores de selección y reproducción (se elimino este operador genético debido a que estaba incorporado en la nueva estrategia de selección), además de la inclusión de nuevos operadores de intensificación y diversificación. En pro de una ejecución a un menor costo computacional, la inclusión de conocimiento experto, y una guía más inteligente sobre el campo de búsqueda de las posibles soluciones.

4.2.4 Implementar el algoritmo propuesto de programación genética, y validar sus bondades por medio de simulación contra series benchmark de la literatura de predicción de series de tiempo, comparando el error de predicción, la inclusión de los rezagos definidos, y la identificación de las componentes de los modelos predicción de series de tiempo

En el Capítulo 2 se plantearon los cambios propuestos al algoritmo original de GP de una manera teórica, mientras que el Anexo C se mostró la implementación y uso del algoritmo propuesto (GP-FB), en un lenguaje abierto, de amplio uso en la comunidad académica como lo es R, de una manera clara, estructurada y eficiente. Adicionalmente, en la Sección 2.3 se validó su funcionalidad y capacidad de recobrar la ecuación (estructura y parámetros) a partir de los datos, por medio de series de tiempo con ecuación de generación conocida;

superando en el error de aproximación (menor medida de error) y en estructura (la solución fue similar a la ecuación generadora) al algoritmo original de Koza [3] y sus principales modificaciones.

Además, en el Capítulo 3 se corroboró la capacidad de predicción del algoritmo propuesto contra los principales modelos de predicción de series de tiempo (ARIMA, ANN, MLP, SVM, GP) presenten en la literatura, validado los resultados de la ejecución de GP-FB en cinco de las series benchmark de predicción de series de tiempo más empleadas en la literatura, superándolos en casi todas las series de acuerdo con las distintas medidas de error de aproximación empleadas.

4.3 Aportes a la investigación

A continuación se describen los distintos tipos de aporte realizados en este trabajo:

4.3.1 Aportes teóricos

Realizando un análisis general de las modificaciones propuestas al algoritmo original de GP consignadas en el Capitulo 2 de este trabajo, se puede identificar los siguientes aportes teóricos a la predicción de series de tiempo utilizando GP:

- Redefinición de los terminales y operadores del algoritmo de GP, eliminando la dependencia de la selección y precisión de las constantes y rezagos empleados. Además de reducir el espacio de búsqueda de los mismos, centrándose en la identificación de estructuras funcionales y no en las perturbaciones del modelo (constantes y parámetros).
- Eliminación del operador genético de clonación por medio de la redefinición del operador de selección, en el cual se selecciona un esquema (μ, λ) permitiendo un equilibrio entre la intensificación y la diversificación de soluciones en el espacio de búsqueda.
- Redefinición del operador genético de cruce permitiendo la incorporación de la información estructural de los padres al hijo en la dirección de mejora en el espacio funcional definido, esto és, una mejora continua del mismo.
- Redefinición del operador genético de mutación que permitiese pequeños cambios en pro de una mejora en el valor de Aptitud de los individuos (más acorde con la definición de algoritmos genéticos).
- La inclusión de conocimiento experto a partir del análisis previo de la serie de tiempo, representado en la limitación del espacio de búsqueda y selección adecuada de las funciones/bloques funcionales a ser utilizados.
- Aumento de la consistencia matemática de la solución encontrada y la parsimonia. Además de una interpretación más cercana a las características del fenómeno vía asociación de modelos existentes.
- Redefinición de la función de aptitud ponderando el error de predicción con el tamaño y complejidad de los individuos.
- Inclusión del operador de intensificación para optimizar los parámetros de la solución encontrada, y reinicios de la población para el logro de la diversificación de los individuos en el espacio de búsqueda, supeditado a las condiciones de la población durante el proceso de búsqueda del algoritmo de GP.

4.3.2 Aportes metodológicos

Entre los aportes metodológicos encontramos que:

- Se realizó un proceso ordenado y reproducible de validación de la literatura actual en predicción de series de tiempo utilizando GP, por medio de la metodología SRL, analizando los distintos aportes de cada autor de acuerdo a los criterios de búsqueda empleados.
- Se desarrolló de un proceso de búsqueda focalizado en zonas de interés por medio de la adecuación de los porcentajes en aplicación de los operadores genéticos, y en la aplicación de sus versiones originales versus las propuestas realizadas en este trabajo.
- Se realizo un proceso de validación con distintos criterios, de una forma ordenada y reproducible, tanto a nivel conceptual (con funciones de generación conocida) como práctico (con series benchmark de la literatura de predicción de series de tiempo) que generan una verificación cuantificable de las mejoras obtenidas por los cambios propuestos.

4.3.3 Aportes prácticos

Durante el proceso de implementación y pruebas de los cambios propuestos al algoritmo de GP en este trabajo se realizaron los siguientes aportes prácticos:

- Una nueva implementación del algoritmo de GP que permite superar los problemas presentes en su aplicación a la predicción de series de tiempo.
- Implementación eficiente de los árboles sintácticos a partir del uso de expresiones regulares y el aprovechamiento del procesamiento matricial del lenguaje.
- Validación de las mejoras propuestas por medio de la simulación contra otras técnicas de inteligencia computacional aplicadas a series benchmark de la literatura.

4.4 Trabajo futuro

Como trabajo futuro se propone el análisis de nuevos operadores, la incorporación de más bloques funcionales —principalmente aquellos que exhiben relaciones no lineales y su influencia en la aproximación de las series temporales—, y la generación de un modelo general de inclusión de información exógena, adicional a la incorporación de los rezagos de la serie temporal como únicas variables de interés.

Extrapolar los hallazgos de esta investigación, tanto a nivel estructural como el uso de bloques funcionales, en el análisis y generación de programas, circuitos y soluciones a problemas generales del ámbito académico y de industria a partir de las distintas versiones posibles de GP.

Generación de modelos de predicción de series de tiempo industriales y económicas de relevancia para el medio, que permitan ampliar su uso con resultados superiores a los suministados por otras técnicas.

A. Anexo: Ampliación modelos de regresión de series de tiempo

A continuación son analizados los principales modelos utilizados en la predicción de series de tiempo, identificando sus respectivos bloques funcionales (los cuales son denotados por B_i, donde i es el número del bloque) a partir de los definidos en la Sección 2.1, demostrando así, que es posible la generación de los principales modelos de regresión para series de tiempo presentes en la literatura a partir de bloques funcionales, brindando la base de la generación de nuevos modelos de manera automática a partir de los mismos.

- **Modelo Autorregresivo (AR)**
  AR (Autoregressive Model) es un modelo estadístico que corresponde al modelo estándar de regresión, en el que se considera que la observación actual (x_t) es producto de los aportes de las p observaciones anteriores (suele ser denotado como $AR(p)$) [13]. En su forma general corresponde a:

$$x_t = \sum_{i=1}^{p} \varphi_i x_{t-i} + \varepsilon_t = B_1 + B_3$$

  donde p se calcula a partir del análisis de correlaciones, ε_t es iid de preferencia $N(0, \sigma^2)$, x_{t-i} es el vector de entradas de **x** para el rezago $t - i$, φ_i es un constante real a encontrar.

  Una de las principales características de los modelos AR es su similitud a una regresión lineal múltiple en la cual las entradas corresponden a rezagos excluyendo el uso del intercepto, siendo absorbido por el termino del error ε_t.

- **Modelo de Media Móvil (MA)**
  MA (Moving Averange Model) es un modelo estadístico en el cual se considera que la observación actual (x_t) es producto del promedio de las q observaciones anteriores (suele ser denotado como $MA(q)$) [13]. En su forma general corresponde a:

$$x_t = \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t = B_2 + B_3$$

  donde q se calcula a partir del análisis de correlaciones parciales, ε_t iid de preferencia $N(0, \sigma^2)$, ε_{t-i} es el vector diferencias para el rezago $t - i$, y φ_i es un constante real a encontrar.

  Se debe tener en cuenta que el ruido blanco $\{\varepsilon_t\}$ es inobservable, y su implementación suele ser más compleja que los modelos AR. La utilidad de los modelos MA está en proveer representaciones parsimonias para series de tiempo que exhiben MA como una estructura de correlación y su teórica tratabilidad dado por el ε_t es iid [13].

- **Modelo ARMA**

ARMA (Autoregressive Moving Averange Model) es un modelo estadístico en el cual se considera que la observación actual (x_t) es producto de una combinación lineal tanto de la interacción de las p observaciones anteriores como del promedio de las q observaciones anteriores (se suele denotar como $ARMA(p,q)$) [13]. En su forma general corresponde a:

$$x_t = \sum_{i=1}^{p} \varphi_i x_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t = B_1 + B_2 + B_3$$

También puede ser escrito en función del operador de rezago L así:

$$\left(1 - \sum_{i=1}^{p} \varphi_i L^i\right) x_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \varepsilon_t = B'_{11} * B_{23} = B'_{12} * B_3$$

donde:

$$L^i = x_{t-i}$$
$$B^i x_t = x_{t-i}$$
$$\theta = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \varphi = \left(1 - \sum_{i=1}^{p} \varphi_i L^i\right)$$
$$B'_{11} = B_{11} \text{ con } r_i = 1 \text{ y } s_i = 0$$
$$B'_{12} = B_{12} \text{ con } u_i = 1 \text{ y } v_i = 0$$

Por lo que es equivalente a: $\varphi x_t = \theta \varepsilon_t$.

Los modelos ARMA es uno de los modelos paramétricos más utilizados para el análisis de series de tiempo, debido a su flexibilidad en la aproximación de procesos estacionarios. Sin embargo, es de anotar que no constituye un aproximador universal dadas sus limitaciones en el modelamiento de fenómenos no lineales.

- **Modelo ARIMA**

ARIMA (Autoregressive Integrated Moving Average) es un modelo estadístico que utiliza variaciones y regresiones de datos [13], con el fin de encontrar patrones para una predicción hacia el futuro; es decir, utiliza datos rezagados para explicar el comportamiento de las variables a futuro. El orden de un modelo ARIMA se nota como (p,d,q) en donde, p es la autor regresión, d es la integración o diferenciación y q es la media móvil. En dicho modelo se hace necesario seleccionar cuales van a ser los argumentos que el modelo identificará posteriormente para dar comienzo al pronóstico de las series de tiempo. En su forma general corresponde a:

$$\left(1 - \sum_{i=1}^{p} \varphi_i L^i\right)(1 - L)^d x_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \varepsilon_t$$

Equivalente a:

$$B'_{11} * B_{13} * B_{23} = B'_{12} * B_3$$

Igualmente:

$$x_t = B_{26}$$

Donde d es calculado a partir del análisis de ciclo. Adicionalmente se puede deducir que un modelo $ARMA(p,q) = ARIMA(p,0,q)$.

- **Modelo Estacional ARIMA (SARIMA)**

  SARIMA (Seasonal Autoregressive Integrated Moving Average Model) es un modelo estadístico equivalente al ARIMA en el cual se incluye la componente de la estacionalidad [13]. En su forma general corresponde a:

$$\Phi_s(B^s)\varphi(B)(1-B)^d(1-B^s)^d X_t = H_s(B^s)h(B)\varepsilon_t$$

  Equivalente a:

$$B_{14} * B_{15} * B'_{16} * B_{16} * B_{23} = B_{17} * B_{18} * B_3$$

  donde:

  $B^i = L^i = x_{t-i}$
  Φ, φ, H, h polinomios con periodo s
  $B'_{16} = B_{16}$ con $s = 1$.

- **Modelo NARX**

  NARX (Nonlinear Autoregressive Exogenous Model) es un modelo no lineal auto regresivo (como extensión del modelo ARX (Autoregressive Exogenous Model)) el cual considera que la observación actual (x_t) es producto tanto de las p observaciones anteriores como de otras variables exógenas U [84]. En su forma general corresponde a:

$$x_t = F\left(x_{t-1}, \dots, x_{t-p}, U_{t-1}, \dots, U_{t-q}\right) + \varepsilon_t$$

  donde $F(.)$ puede ser lineal o no lineal, x representa los rezagos, y $U_{t-\mathrm{j}}$ las variables externas al modelo. Dependiendo de la forma de $F(.)$ se generarían sus bloques funcionales.

- **Modelo de Vector de Autoregresión (VAR)**

  VAR (Vector Autoregression Model) es un modelo econométrico usado para capturar la evolución e interdependencias entre múltiples series de tiempo [85]. En su forma general corresponde a:

$$x_t = c + A_1 x_{t-1} + \cdots + A_p x_{t-p} + \varepsilon_t$$

  Equivalente a:

$$x_t = c + \sum_{i=1}^{p} A_i x_{t-i} + \varepsilon_t = B_7 + B_1 + B_3$$

  donde se puede deducir que el modelo VAR es equivalente al AR (p) con coeficientes A_i matriciales.

- **Modelo Autorregresivo de Umbrales (TAR)**

  TAR (Threshold Autoregressive Model) es un modelo econométrico motivado principalmente por las características no lineales comunes observadas y la asimetría en la disminución y aumento de patrones en procesos económicos. En los modelos TAR es utilizado un umbral en el espacio para intentar realizar una aproximación lineal [86, 87]. En su forma más general corresponde a:

$$x_t = \sum_{i=1}^{n+1} \gamma_i g_i(x_{t-1}) = B_9$$

donde:

$$\gamma_1 = \begin{cases} 1 \; si \; X_{t-1} \leq Umbral_1 \\ 0 \; en \; otro \; caso \end{cases}$$

$$\gamma_i = \begin{cases} 1 \; si \; X_{t-1} \leq Umbral_i \; y \; \gamma_j = 0 \; para \; \forall_{j<i} \\ 0 \; en \; otro \; caso \end{cases}$$

Equivalente a:

$$X_t = \begin{cases} g_i(X_{t-1}) \; si \; X_{t-1} \leq Umbral_1 \\ \vdots \\ g_i(X_{t-1}) \; si \; Umbral_{i-1} < X_{t-1} \leq Umbral_i \end{cases}$$

Los cuales dependiendo del número de umbrales se suelen denotar como $TAR(n)$ donde n es el número total de umbrales (cortes) de la función [87]. Por ejemplo, un modelo $TAR(1)$ puede ser de la forma:

$$X_t = \begin{cases} -1 + X_{t-1} \; si \; X_{t-1} \leq 0 \\ -X_{t-1} \; si \; X_{t-1} > 0 \end{cases}$$

donde $a_t \; es \; iid \; N(0,1)$.

- **Modelo Autorregresivo de Umbrales Auto Excitado (SETAR)**

  SETAR (Self-Exciting Threshold Autoregressive Model) es un modelo equivalente al TAR en el cual el valor de salida no depende solo del rezago anterior (x_{t-1}) sino de una serie de rezagos $i = \{1,2, \dots, p\}$ [86], tal que satisfacen:

$$x_t = \phi_0^{(j)} + \phi_1^{(j)} x_{t-1} - \cdots - \phi_p^{(j)} x_{t-p} + a_t^{(j)}$$

Equivalente a:

$$x_t = \sum_{i=1}^{p} \phi_i^{(j)} x_{t-p} + a_t^{(j)} = B_1 + B_7$$

Siempre que se cumpla: $\gamma_{j-1} \leq X_{t-d} \leq \gamma_j$. Donde $k, d, j = \{1, \dots, k\}$ son enteros positivos, γ_i son números reales tal que: $-\infty = \gamma_0 < \gamma_1 < \cdots < \gamma_{k-1} < \gamma_k = \infty$.

El superíndice (j) significa el régimen $\{a^{(j)}\}$ los cuales son secuencias iid con media 0 y varianza σ_j^2 y mutuamente diferentes para distintos j. El parámetro d se refiere a al rezago y γ_j a los umbrales. Es decir, se tienen distintos modelos autorregresivos determinados por los umbrales [86, 88-93].

- **Modelo Autorregresivo de Heteroelasticidad Condicional (ARCH)**
  ARCH (Autoregressive Conditional Heteroskedasticity Model) es un modelo econométrico el cual considera la varianza del término actual de error (innovación) es producto de los p anteriores términos [94]. En su forma general corresponde a:

$$x_t = \sigma_t \varepsilon_t = B_6'$$

$$\sigma_t^2 = c + \sum_{i=1}^{p} b_i x_{t-i}^2 = B_7 + B_1''$$

donde:

$$B_6' = B_6 \quad \text{con} \quad \sigma_t^2 = \sigma_t$$
$$B_1'' = B_1 \quad \text{con } x_{t-i} = x_{t-i}^2$$

- **Modelo Autorregresivo Generalizado de Heteroelasticidad Condicional (GARCH)**
  GARCH (Generalized Autoregressive Conditional Heteroskedasticity Model) es un modelo econométrico el cual considera la varianza del término actual de error (innovación) es producto de los p anteriores términos, adicional a la contribución de las observaciones anteriores [95]. En su forma general corresponde a:

$$x_t = \sigma_t \varepsilon_t = B_6'$$

$$\sigma_t^2 = c + \sum_{i=1}^{p} b_i x_{t-i}^2 + \sum_{i=1}^{q} a_i \sigma_{t-i}^2 = B_7 + B_1'' + B_2''$$

donde:

$$B_6' = B_6 \text{ con } \sigma_t^2 = \sigma_t$$
$$B_1'' = B_1 \text{ con } X_{t-i} = X_{t-i}^2$$
$$B_2'' = B_2 \text{ con } \sigma_{t-i} = \sigma_{t-i}^2,$$

Es de resaltar que p es hallado a partir del análisis de la matriz de auto correlación de ε_t.

- **Modelo de Regresión Polinomial Local Multivariado (MLP)**
  MLP (Multivariate Local Polynomial Regression Model) es un modelo estadístico basado en los modelos de kernel el cual corresponde a una función de densidad de probabilidad $k(x)$ [96], tal que:

$$k(x) \geq 0$$
$$\int k(z)dz = 1$$

En su forma general corresponde a:

$$\int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty} k(x)dx = 1$$

donde $k(x)$ corresponde a la función de kernel, la cual puede ser de la forma:

$$\mathrm{k(x)} = \begin{cases} \prod_{\mathrm{i=1}}^{\mathrm{p}} \mathrm{k(x_i)} = \mathrm{B_{22}} \\ \mathrm{C_{k,p}k(\|x\|)} = \mathrm{B_{21}} \\ 2\pi^{-\frac{\mathrm{p}}{2}} \mathrm{e}^{-\frac{\|\mathrm{x}^2\|}{2}} = \mathrm{B_{20}} \end{cases}$$

El cual es optimizado por medio de:

$$\int \mathrm{x_i k(x)dx} = 0$$

Que para el caso discreto y teniendo en cuenta una función de kernel de la forma:

$$\mathrm{k(x_i)} = \varphi_{\mathrm{i}}$$

Corresponde a un modelo $AR(p)$.

- **Modelo Funcional Autorregresivo (FAR)**
  FAR (Functional Autoregressive Model) es un modelo estadístico, en el cual la observación actual está determinada por una relación no lineal entre la observación anterior y una base X_{t-d} [97]. En su forma general corresponde a:

$$X_t = \sum_{i=1}^{p} a_i X_{t-d} X_{t-i} + X_{t-d}\sigma\varepsilon_t = 1 - B''_{11} + B'_8 * \mathrm{B'_6}$$

donde:

$$B'_8 = B_8 \text{ con } \Delta = 0, s = 1$$
$$B''_{11} = B_{11} \text{ con } r_i = 0, s_i = 1, h = d$$

El cual se puede deducir del modelo ARMA general.

- **Modelo Exponencial Autorregresivo (EXPAR)**
  EXPAR (Exponential Functional Autoregressive Model) es un modelo estadístico equivalente a los modelos FAR utilizando términos exponenciales [98]. En su forma general corresponde a:

$$x_t = \sum_{i=1}^{p} (\alpha_1 + \{\beta_i + \Upsilon_i x_{t-d}\} e^{-p_i X_{t-d}^2}) x_{t-i} + \varepsilon_t = \sum_{i=1}^{p} \alpha_1 x_{t-i} + \sum_{i=1}^{p} \{\beta_i + \Upsilon_i x_{t-d}\} e^{-p_i X_{t-d}^2} x_{t-i} + \varepsilon_t$$

Equivalente a:

$$X_t = B_1'' + 1 - B_{11}''' + B_3$$

donde:

$B_1'' = B_1$ con $\varphi_i = \alpha_1$
$B_{11}''' = B_{11}$ con $\varphi_i = B_{19}, r_i = \beta_i, s_i = \Upsilon_i$

- **Modelo Funcional de Coeficiente Adaptativos Autorregresivo (AFAR)**
AFAR (Adaptive Functional Coefficient Autoregressive Model) es un modelo estadístico generalizado basado en los modelos FAR [99]. En su forma general corresponde a:

$$x_t = g_0(\beta^T x_{t-1}) + \sum_{i=1}^{p} g_i(\beta^T x_{t-1}) x_{t-i} + \varepsilon_t = B_{24}' + B_1''' + B_3$$

donde ε_t es independiente de x_{t-1}, β^T constituye la dirección del modelo dependiente permitiendo implementaciones más robustas del modelo sin implicar un aumento considerable de complejidad, $B_{24}' = B_{24}$ con $i = 0$ y $d = 1$, $B_1''' = B_1$ con $\varphi_i = B_{24}$, y $d = 1$.

Los modelos AFAR surgieron a raíz de que los modelos FAR tienen una alta dependencia a selección de la variable dependiente x_{t-d}, lo cual es un factor que limita sus posibles aplicaciones. Por lo que permite una combinación lineal de los valores de la variable independiente.

- **Modelo Bilineal (BL)**
El modelo Bilinear (BL – Bilinear Model) se basa en la aproximación de la no linealidad de una serie de tiempo por medio de la expansión de segundo orden de la serie de Taylor sobre la función:

$$x_t = f(a_t, a_{t-1}, \dots)$$

Lo que genera como resultado:

$$x_t = c + \sum_{i=1}^{p} \varphi_i x_{t-i} - \sum_{j=1}^{q} \theta_j a_{t-j} + \sum_{i=1}^{m} \sum_{j=1}^{s} \beta_{ij} x_{t-i} a_{t-j} + a_t$$

donde p, q, m, s son enteros no negativos mayores o iguales a 1. Este modelo fue introducido por Grander & Andersen [100] quienes lo han investigado arduamente. Subba Rao & Gabr discutieron sus respectivas propiedades y aplicaciones [101], y Liu & Brockwell estudiaron los modelos bilineales generales [102]. Las propiedades de los modelos bilineales como las condiciones de estacionalidad han sido derivadas por medio de: a) La colocación del modelo en forma estado-espacio. b) La utilizando de la ecuación de transición de estados para expresar el estado como un producto de las innovaciones pasadas y un vector de coeficientes aleatorios.

De acuerdo con lo anterior y lo trabajado por [100], se puede deducir que la observación actual es una relación no lineal entre el error actual y los errores de las observaciones anteriores [103], cuya expresión general es de la forma:

$$x_t = \mu + \sum_{i=1}^{q} \beta_i \varepsilon_{t-i} \varepsilon_t + \varepsilon_t = B_7 + B_5'' + B_3$$

donde $B_5'' = V_5$ con $u_i = 0$, y $v_i = 1$; cuyos primeros dos momentos condicionales son de la forma:

$$E(X_t|F_{t-1}) = \mu$$

$$Var(X_t|F_{t-1}) = \left(1 + \sum_{i=1}^{s} \beta_i \varepsilon_{t-i}\right)^2 \sigma_\varepsilon^2$$

- **Modelo STAR**

STAR (Smooth Transition Autoregressive Model) es un modelo estadístico, el cual surgió como respuesta a las críticas sobre los modelos SETAR los cuales son esencialmente condicionales, lo que significa que constituye una ecuación discontinua en el espacio de aproximación; los puntos de discontinuidad son llamados Umbrales denotados como $\{\gamma_j\}$. De acuerdo a esto, fueron propuestos los modelos Smooth TAR [104], los cuales en general corresponden a modelos AR en el cual se pondera la función de transición $F(.)$, generando un modelo general de la forma:

$$x_t = c_0 + \sum_{i=1}^{p} \varphi_i x_{t-i} + F\left(\frac{x_{t-d} - \Delta}{s}\right)\left(c_1 + \sum_{i=1}^{p} \theta_i x_{t-i}\right) + \varepsilon_t$$

Equivalente a:

$$X_t = B_7 + B_1 + B_8 * (B_7 + B_1) + B_3$$

donde $F(.)$ es la función de transición (puede ser logística, exponencial, distribución acumulativa, entre otras), d es el retardo máximo a tenerse en cuenta, s es la escala del modelo, y Δ es la posición del modelo de transición [105].

De acuerdo con lo anterior, la media condicional de un modelo STAR es una combinación linear ponderada de las siguientes ecuaciones:

$$\mu_{1t} = c_0 + \sum_{i=1}^{p} \varphi_i x_{t-i}$$

$$\mu_{2t} = (c_0 + c_1) + \sum_{i=1}^{p} (\varphi_i + \theta_i) x_{t-i}$$

Cuyos pesos son determinados por medio de $F(.)$. Además, un prerrequisito para la estacionalidad de un modelo STAR es que todos los ceros (0) de los dos modelos auto regresivos polinomiales estén fuera del circulo unitario.

La diferencia entre un TAR y un STAR radica en la diferenciación de su función de media, aunque en la práctica, hallar tanto Δ como s resulta demasiado difícil, por lo que en la práctica se suele tomar valores cercanos a 1 [106].

- **Modelo de Lógica Difusa (FUZZY)**

FUZZY (Fuzzy Model) es un modelo de inteligencia computacional el cual considera que existen infinitos valores entre el $falso$ (0) y $verdadero$ (1) (tradicionales de la lógica de predicados), de acuerdo a esto y dado que un modelo TAR es básicamente un conjunto de modelos AR locales; y considerando que un

modelo fuzzy está compuesto por un conjunto de reglas fuzzy, y cada regla fuzzy tiene correspondencia con un modelo AR [107], el modelo Fuzzy se reduce a una extensión del modelo STAR.

- **Red Neuronal Autoregresiva (ARNN)**

Una red neuronal (Neural Network) tiene como objetivo simular el sistema nervioso de los animales, es decir, simular un conjunto de redes neuronales a través de instrucciones matemáticas, las cuales se encuentran agrupadas en capas; dichas capas están constituidas por una capa de entrada, una o más capas ocultas y una capa de salida; en dichas capas hay neuronas que han sido entrenadas y poseen una función matemática denominada función de transferencia, que genera la señal de salida de la neurona a partir de señales de entrada. Las ARNN (Autoregressive Neural Network) es una extensión de las NN en las cuales se les adiciona un término lineal [108-109]. En su forma general corresponde a:

$$x_t = \beta_0 + \sum_{i=1}^{p} \varphi_i x_{t-i} + \sum_{h=1}^{H} \beta_h G\left((2\sigma_t)^{-1} \alpha_{0h} + \sum_{i=1}^{I} \alpha_{ih} x_{t-i} \right) + \varepsilon_t$$

Equivalente a:

$$x_t = B_7 + B_1 + B_{10} + B_3.$$

donde $G(.)$ es la función de activación de las neuronas de la capa oculta, σ es la desviación estándar de los errores, ε_t es una variable aleatoria que sigue una distribución normal estándar, H es el número de neuronas en la capa oculta, I es el número de regresores, y σ es la desviación estándar de x_t; su uso evita tener que transformar x_t para restringir sus valores al rango de la función $G(.)$. Cabe anotar que una NN es una ARNN con $\varphi_i = 0$.

Los parámetros del modelo ($\boldsymbol{\Omega} = [\beta *, \beta_h, \varphi_i, \alpha *, h, \alpha_{ih}]$, para $h = \{1, \dots, H\}$; $i = \{1, \dots, I\}$) son obtenidos maximizando el logaritmo de la función de verosimilitud de los errores:

$$L = -\frac{T}{2} log(2\pi) - \frac{T}{2} log(\sigma^2) - \frac{1}{2} \sum_{t=1}^{T} \frac{\varepsilon^2}{\sigma^2}$$

Mediante alguna técnica de optimización, usualmente basada en gradientes. σ^2 es el promedio de los errores al cuadrado. La ecuación L es obtenida al asumir que los residuales ε_t siguen una distribución normal con media cero y varianza desconocida; la maximización de la función de verosimilitud equivale a minimizar el error cuadrático medio, que es el procedimiento común en la literatura de redes neuronales.

Se propone la utilización de funciones tipo sigmoidea ya que algunos autores han sugerido por su experiencia práctica, que este tipo de funciones que son simétricas alrededor del origen convergen más rápidamente que la función sigmoidea tradicional; adicionalmente, la adición de un término lineal puede ayudar a la convergencia, ya que se evita la saturación de la neurona o unidad de procesamiento en la capa oculta, y garantiza un gradiente mínimo cuando la salida neta de la función sigmoidea es cercana a sus valores extremos. Consecuentemente con las razones expuestas, $G(.)$ es especificada como:

$$G(u) = \frac{1 - exp(-u)}{1 + exp(-u)} + ku$$

donde κ es una constante pequeña. Vale resaltar que, para un conjunto cualquiera de parámetros $\boldsymbol{\Omega}$, los parámetros β_h pueden restringirse a ser positivos, ya que la contribución neta de cada unidad oculta $\beta_h G(.)$ no cambia de signo si los parámetros β_h, $\alpha *, h$, α_i, h cambian de signo, puesto que:

$\beta_h G(u) = -\beta_h\, G(-u)$. Igualmente, para evitar la multiplicidad de configuraciones en la capa oculta se puede obligar a que los parámetros β_h estén siempre ordenados de forma creciente: $0 < \beta_1 < \beta_2 < \ldots < \beta_h$. Al imponer que los parámetros φ_i sean igual cero, el modelo ARNN se reduce a un perceptron multicapa (MLP). Si H se hace igual a cero, x_t se reduce a un modelo Autorregresivo lineal con entradas exógenas (ARX).

- **Modelo General ARMA**

De acuerdo con los modelos anteriores y analizando las posibilidades de extensión de los modelos ARMA es posible deducir un modelo ARMA general de la forma $ARMA(p, q, h, k, r, s, u, v)$:

$$X_t = c + \sum_{i=1}^{p} \{\varphi_i X_{t-i}(r_i + s_i X_{t-h})\} + \sum_{i=1}^{q} \{\theta_i \varepsilon_{t-i}(u_i + v_i \varepsilon_{t-k})\} + \sigma^2 \varepsilon_t$$

Y apartir de uso del operador de rezago L, es equivalente a:

$$\left(1 - \sum_{i=1}^{p} \varphi_i L^i (r_i + s_i L^h)\right) X_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\, (u_i + v_i L^k)\right) \sigma^2 \varepsilon_t + c$$

Igual a:

$$B_{10} * B_{23} = B_{11} * B_6 + B_7$$

Considerando el hecho que los parámetros son variables mudas, es posible simplificar la expresión a:

$$x_t = c + \sum_{i=1}^{p} \{\varphi_i x_{t-i} r_i + \varphi_i x_{t-i} s_i x_{t-h}\} + \sum_{i=1}^{q} \{\theta_t \varepsilon_{t-i} u_i + \theta_i \varepsilon_{t-i} v_i \varepsilon_{t-k}\} + \sigma^2 \varepsilon_t$$

Entonces:

$$x_t = c + \sum_{i=1}^{p} \varphi_i x_{t-i} \sum_{i=1}^{p} s_i x_{t-i} x_{t-h} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \sum_{i=1}^{q} v_i \varepsilon_{t-i} \varepsilon_{t-k} + \sigma^2 \varepsilon_t$$

De donde se puede apreciar que existen distintas maneras de representar la misma expresión de acuerdo con el nivel de simplificación y bloques funcionales a ser utilizados.

- **Modelo General ARIMA**

De acuerdo con los modelos anteriores y analizando las posibilidades de extensión de los modelos ARIMA es posible deducir un modelo ARIMA general de la forma $ARIMA(p, d, q, h, k, r, s, u, v)$:

$$\left(1 - \sum_{i=1}^{p} \varphi_i L^i (r_i + s_i L^h)\right) (1 - L)^d x_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\, (u_i + v_i L^k)\right) \sigma^2 \varepsilon_t + c$$

Equivalente a:

$$B_{10} * B_{13} * B_{23} = B_{11} * B_6 + B_7$$

donde un modelo ARIMA corresponde a un modelo ARIMA general con: $r = u = 1$ y $s = v = c = 0$.

- **Modelo General de una Red Neuronal**

  De acuerdo con los modelos anteriores y analizando las posibilidades de extensión de las componentes de una red neuronal es posible deducir una forma general para las NN dada por la expresión:

$$X_t = \beta_0 + F\left(\frac{X_{t-d} - \Delta}{s}\right)\sum_{i=1}^{p} \varphi_i K(X_{t-i}) + \sum_{h=1}^{H} \beta_h G\left((2\sigma_t)^{-1}\alpha_{0h} + \sum_{i=1}^{I} \alpha_{ih} X_{t-i}\right) + \varepsilon_t$$

  Equivalente a:

$$x_t = B_7 + B_8 * B_9 + B_{10} + B_3$$

  donde $F(.)$ es la función de transición de un modelo tipo STAR.

B. Anexo: Programación Genética

La programación genética (GP, por su sigla en inglés) es la generalización de los algoritmos genéticos, la cual busca determinar el mejor modelo que aproxime las relaciones entre una variable de entrada y una o más variables de salida; en este contexto se suele llamar regresión simbólica (SR, por su sigla en inglés). Esta técnica se basa en el uso de individuos (ecuaciones matemáticas representadas en forma de árbol) y operadores genéticos (cruce y clonación) para hallar el óptimo (mínimo) global de una función de error dada [3, 14].

A continuación se realiza una revisión de los principales componentes y la descripción del algoritmo general definido por Koza [3].

Algoritmo básico

El algoritmo de GP es muy similar al de los algoritmos genéticos (GA, por su sigla en inglés), en el que para GP los operadores genéticos (cruce y clonación) son aplicados a los arboles (individuos) los cuales representan segmentos de código o ecuaciones (para el caso de la SR) y no a valores específicos de una función como en GA (busca el óptimo de la función objetivo). De acuerdo con Koza [3], los pasos que describen el algoritmo original de GP son:

1. En la iteración o generación inicial ($g = 0$) se crea una población (P_0) de n individuos.
2. Para cada individuo S_i, con $i = \{1, \ldots, n\}$, se evalúa la función de aptitud $f(S_i)$.
3. Se genera una nueva población P_g^* aplicando los operadores genéticos a la población actual P_g, de la siguiente forma:
   a) Son seleccionados los padres a los cuales se le aplicará los operadores de cruce y clonación de una manera probabilística.
   b) A diferencia de los algoritmos genéticos tradicionales no se considera el operador de mutación por lo que se limita a cruce y clonación.
   c) La población de hijos P_g^* Se reemplaza la población actual P_g así: $P_g = P_g^*$.
4. Se evalúan los criterios de parada (usualmente es utilizado el número máximo de generaciones), sino se cumplen se vuelve al paso 2. En caso contrario se termina la ejecución del algoritmo.

En las siguientes secciones son detallados los principales elementos del algoritmo.

Componentes del algoritmo

A continuación se describen los principales componentes del algoritmo de GP sobre los cuales se aplica los pasos del mismo:

- **Terminales**
  Corresponden a los argumentos base de las funciones, entre los cuales se encuentran el vector de variables de entrada (puede ser un vector de vectores - matriz) que suele ser llamado X y los coeficientes a ser utilizados (parámetros - constantes) [3].

- **Operadores o funcionales**
  Son todas aquellas funciones que pueden ser aplicadas a los nodos terminales; por ejemplo, los elementos del conjunto de operadores aritméticos básicos $F = \{+, -, *, /\}$, los cuales poseen atributos como: símbolo, nombre, tipo y número de argumentos [3, 14]. Pueden ser representados funcionalmente como:

  - Operador suma: {símbolo: +; número de argumentos: 2; tipos de argumentos: [numérico, numérico]}.
  - Operador sustracción: {símbolo: -; número de argumentos: 2; tipos de argumentos: [numérico, numérico]}.
  - Operador multiplicación: {símbolo: *; número de argumentos: 2; tipos de argumentos: [numérico, numérico]}.
  - Operador división: {símbolo: /; número de argumentos: 2; tipos de argumentos: [numérico, numérico]}.

  Se debe tener en cuenta que los argumentos pueden ser el resultado de la evaluación de otros operadores (subárbol) [14].

- **Individuo (árbol)**
  Es la combinación de operadores y terminales organizados de manera jerárquica (suele ser árbol) equivalente a una ecuación matemática; por ejemplo, sea la ecuación: $G(x, y) = (\mathrm{x} - y) + \frac{x}{y^2}$, cuyos componentes son: $F = Operadores = \{+, -, *, /\}, T = Terminales = \{x, y\}$.

  Se debe tener en cuenta que en la representación de ecuaciones matemáticas a partir de árboles jerárquicos compuestos por operadores y terminales, la generación del mismo es en sentido post-orden (todo nodo padre es evaluado después de su subárbol izquierdo y derecho) y su expresión computacional dependerá del tipo de sistema utilizado (procedural, matemático, declarativo, híbrido) [3, 14].

  Adicionalmente, tanto los operadores como los terminales de un individuo deben cumplir las siguientes propiedades:

  - *Propiedad de Cierre*
    Cada función debe ser procesada para todos los posibles valores de los argumentos (resultados de la evaluación de otros operadores o terminales), esto implica que las funciones asociadas a los operadores susceptibles a valores erróneos deben ser redefinidas para superar, por ejemplo: divisiones por cero, logaritmos negativos, entre otros [14].

  - *Propiedad de Suficiencia*
    Los operadores y terminales seleccionados deben ser capaces de generar aquella función que mejor describa los datos, esto se logra al introducir al menos una función no lineal (dado que de acuerdo con Runge-Kutta, cualquier función puede escribirse como una combinación de una función no lineal) el problema radica en la selección de los terminales y el tiempo computacional empleado para ello [14].

- **Población**
  Corresponde a un conjunto de n individuos $P(g)$ en una generación específica g. Funcionalmente constituye un vector con cada uno de los individuos de g [3].

- **Generación**
  Corresponde a la iteración específica g del algoritmo, con $g = \{1, 2, \dots, G\}$ y su población específica $P(g)$ [3].

- **Operadores genéticos**
  Los operadores genéticos constituyen la base del algoritmo de GP y son los encargados de la exploración en el campo de búsqueda, permitiendo la adaptación de los individuos y escapar de óptimos locales por medio de la diversificación; en esta sección se analizan los operadores de clonación y cruce definidos en el algoritmo original de Koza [3]:

  - *Clonación (Reproducción)*
    Consiste en seleccionar un individuo y generar una copia idéntica de él; la selección del individuo se basa en el valor de la función de aptitud de los individuos, aplicando alguna de las siguientes técnicas: proporcional a la función de aptitud, selección por rango o selección por torneo [3].

  - *Cruce*
    Consiste en seleccionar dos individuos aleatoriamente de forma proporcional a su aptitud [3]. Para cada uno de los individuos, se selecciona un nodo de cruce aleatorio a partir del cual se intercambian los respectivos sub-arboles [14].

- **Función de aptitud**
  La función de aptitud corresponde a la función sobre la cual se analizará que tan bueno es un modelo (individuo) frente a otro. Para cada uno de los individuos es posible evaluar la función de aptitud (fitness) de distintas maneras, algunas de las propuestas realizadas por Koza [3] han sido:

  - Función de Aptitud Cruda: En la cual se suman los errores por fuera del rango de aptitud permitido (valores distintos al esperado).
  - Función de Aptitud Estandarizada: Es el valor absoluto de la diferencia entre el valor esperado y el valor de evaluar el individuo S_i para el vector de datos de entrada X: $\|f(S_i) - \hat{y}\|$.
  - Función de Aptitud Ajustada: Corresponde a la Función de Aptitud Estandarizada re-escalado entre 0 y 1, donde 0 equivale a aquel individuo con menor valor de función y 1 el de mayor, así: $1 - \frac{\|f(S_i) - y\|}{\sum_{j=1}^{n} \|f(S_j) - y\|}$. Con y igual al resultado esperado.

  Cabe recordar que el algoritmo de GP explorará en el universo de posibles individuos que posean el mejor valor de función de aptitud (indirectamente constituye el de menor error). Para realizar el proceso de exploración, la GP reinterpreta las distintas ecuaciones (modelos) matemáticas, asignándoles a cada uno de ellos individuos (relación de terminales por medio del uso de operadores) estableciendo para ello un algoritmo genérico de aplicación.

- **Mejoras al algoritmo básico**
  Para el algoritmo de GP se han propuesto en la literatura algunas adaptaciones que buscan una mayor estabilidad, tiempos de ejecución más cortos y una mejor aproximación a los datos [12]. Entre ellas se encuentran:

Simplificación de cada uno de los individuos después del paso 5, de acuerdo a reglas de simplificación matemáticas de polinomios, entre las que se encuentran:

- Operadores redundantes, por ejemplo: $x + x + x - x$ que es equivalente a $3 * x$.
- Operaciones básicas entre terminales: $0.1 + 0.1 + 0.1$ que es equivalente a 0.3.

Esta adaptación permite simplificar el árbol resultante, pero limita el espacio de aplicación de los operadores de cruce y mutación, debido a que en estos casos se pasa de poder aplicar los operadores de cruce y mutación de 7 nodos a solo 3: $x + x + x - x = 3 * x$.

Adicional a cambiar totalmente el individuo sin mantener parte de su conocimiento heredado, por ejemplo, en el caso de la simplificación a un solo terminal para el operador de mutación: $0.1 + 0.1 + 0.1 = 0.3$

Utilización de criterios de parada como:

- Máximo número de iteraciones/generaciones G [3].
- Falta de mejora (o muy pequeña a partir de un valor dado de diferencia mínima épsilon) de la función de aptitud (medida de mejora del error determinada por la función de aproximación especifica del individuo) del mejor individuo entre las generaciones g y $g - 1$ [14].
- La falta de mejora es una de las adaptaciones más utilizadas en las implementaciones de GP, dado que a menor error, mayor costo computacional de evaluación, debido principalmente al número de generaciones e individuos necesarios para examinar una mayor porción del espacio de búsqueda y, por ende, hallar más soluciones que puedan disminuir el error.

Adicionalmente, se debe tener en cuenta que la GP es una técnica no determinista, por lo que no es posible garantizar su convergencia en pocas iteraciones y dependerá en gran medida de la inicialización de los individuos y la forma de la serie de datos; tradicionalmente y de acuerdo con las necesidades de análisis se suele seleccionar los valores de $G \geq 100$ y épsilon (diferencia entre los valores de la función de aptitud del mejor individuo de la generación actual y el mejor individuo de la generación anterior: $\|Imenjor_g - Imenjor_{g-1}\|) \leq 0.05$ [3, 14], lo anterior permite que el algoritmo itere hasta converger a un valor de aptitud esperado para la predicción.

Adicionalmente se han introducido dos nuevos operadores genéticos:

- *Mutación*
  Consiste en seleccionar aleatoriamente un nodo (función o terminal) y mutarlo de acuerdo a alguno de los siguientes métodos: Cambiar el nodo aleatoriamente por otra función y/o terminal el cual acepte el mismo número de argumentos [3]; o cambiar todo el subárbol a partir del nodo seleccionado (aleatoriamente) por otro generado aleatoriamente [14].

- *Permutación*
  Consiste en seleccionar aleatoriamente dos nodos del individuo los cuales sean compatibles en número de argumentos y se intercambian entre sí, generando un nuevo individuo permutado. Puede considerarse como un caso particular de la mutación en la cual se mutan dos nodos al tiempo y para cada uno de ellos se selecciona el operador/terminal del otro [12].

De acuerdo con las características de los datos y el comportamiento del algoritmo se puede optar por una u otra mejora teniendo presente las implicaciones de utilización de cada una de ellas [14].

C. Anexo: Paquete gpToolFB, una implementación de programación genética al pronóstico de series de tiempo utilizando bloques funcionales

De acuerdo con el Capítulo 2, es posible definir los bloques funcionales como constituyentes de los terminales de los individuos de GP en grupos de acuerdo con las componentes de la serie temporal (Ciclo, Tendencia, Estacionalidad, Error) adicional a las modificaciones a los operadores genéticos para la focalización de zonas de interés en el proceso de exploración del algoritmo original de GP; a partir de ello, en este anexo, se definirán las distintas estructuras de datos (clases) necesarias para modelarlo, las principales funciones del paquete gpTool, sus argumentos de entrada y valores de salida.

El lenguaje R

R es una suite de programas que facilita la manipulación, cálculo y graficación de funciones y datos. Una de sus principales características es que posee su propio lenguaje de programación llamado "S" el cual incluye operadores clásicos de iteración (loops, while, for, do) adicional a la posibilidad de crear funciones propias de usuario y clases (variables y funciones agrupadas en una entidad) tanto de tipo S3 como S4 [128], siendo esta última más orientada a objetos (las funciones y atributos son encapsulados totalmente en una entidad).

Algunas de las razones por las cuales se eligió R para implementar la GP y su caso particular SR son:

- R es un lenguaje en el cual las matrices y los vectores son tipos de datos nativos, por lo que es posible realizar operaciones aritméticas matricialmente de una manera transparente para el investigador (no es necesario redefinir las operaciones para que operen sobre matrices), lo que permite un manejo más fácil de la estructura del algoritmo y aumento en la velocidad de procesamiento.
- R es un lenguaje de código abierto reconocido en el medio académico, tanto para estadística como para inteligencia computacional, con su propio esquema de distribución y colaboración estándar.
- En R, es realizado automáticamente el manejo (creación y borrado) de memoria de una manera eficiente, lo que reduce considerablemente los tiempos de verificación (es inherente al lenguaje y libera al programador de las tareas de asignación y liberación de memoria física.
- R es un lenguaje interpretado que permite definir en tiempo de ejecución nuevas funciones, clases u objetos, adicional a la evaluación de expresiones (propias del lenguaje y matemáticas en general) a partir de una cadena de texto o archivo (dependiendo del tamaño de la expresión).
- R contiene un gran conjunto de paquetes desarrollados tanto para estadística como inteligencia computacional de libre acceso y fácil instalación de acuerdo con los registros CRAN (Comprehensive R Archive Network).
- R es un lenguaje de orientación híbrida, en el cual es posible trabajar de manera objetual y funcional, lo que permite un esquema de desarrollo amigable al programador.

En este capítulo se mostrará el uso del paquete gpToolFB en el cual los individuos de GP fueron implementados por medio de listas, debido principalmente a los siguientes aspectos:

- Permite disminuir los tiempos de ejecución del algoritmo dado que se manejan expresiones en R cuya evaluación es de manera automática.
- Presenta funciones más limpias y depuradas que permiten su actualización, mantenimiento y entendimiento para el usuario final de una manera más fácil y directa.

A continuación es mostrada la implementación de los distintos componentes de los individuos y algoritmo modificado de GP propuesto en el Capítulo 2 (GP-FB):

Terminales

Los terminales pueden ser una constante o un bloque funcional (teniendo en cuenta que **x** es el $Bloque_0$), por lo que a continuación se muestran los vectores de parámetros (*params*) y parámetros externos (*extParams*) de los bloques funcionales analizados anteriormente, teniendo en cuenta que $\emptyset$ es equivalente a vacio (*null*), por ejemplo es posible definir un bloque así:

$$Bloque_1\left(\sum_{i=1}^{p}\varphi_i x_{t-i}\right):\begin{cases} params = \{\varphi_i\} \\ extParams = \{\mathrm{p}\} \end{cases}$$

$$Bloque_2\left(\sum_{i=1}^{q}\theta_i \varepsilon_{t-i}\right):\begin{cases} params = \{\theta_i\} \\ extParams = \{\mathrm{q}\} \end{cases}$$

Para los cuales sus respectivas funciones son de la forma:

```
nombre <- function(x = NULL, params = NULL, extParams = NULL, snPrint = FALSE,
                   snName = FALSE, snCountParam = FALSE, ...) {
    # Desarrollo de la función
    ...
}
```

Dónde:

- *nombre* corresponde al nombre propio de la función (suele ser nemotécnica, por ejemplo *gpBlock_0* para el bloque funcional 0) .
- x es el vector de entradas para X_{t-i}.
- *params* son los parámetros enviados para su ejecución (si aplica).
- *extParams* son los parámetros externos (si aplica).
- *snPrint* indica si la función retornará una cadena de caracteres con la representación de la función en forma comprensible para el usuario (suele ser el nombre propio de la función utilizada).
- *snName* indica si la función retornará una cadena de caracteres con su representación como expresión valida en R para ser evaluada.
- *snCountParams* indica si la función debe retornar el número total de parámetros que utiliza.

Se debe tener en cuenta que existe un vector de parámetros globales para todos los nodos de un individuo (árbol – expresión matemática), además de las variables adicionales necesarias para su ejecución por medio '…'. Si los indicadores *snPrint, snName* y *snCountParam* son *FALSE*, la función retornará el resultado de ser evaluada para un $x = \mathbf{x}_{t-i}$ bajo *params* y *extParams*. De acuerdo con lo anterior es posible definir nuevos

terminales que cumplan con la estructura básica antes indicada. Si alguno de los argumentos es omitido y es necesario para la correcta ejecución, la función retornará cero (0).

De acuerdo a lo anterior, si el parámetro *snPrint = TRUE* el sistema retorna el valor "gpBlock_1", el cual es la representación en forma de ecuación (fácilmente entendible para el usuario) que se definió para el bloque funcional. Adicionalmente, si el parámetro *snName = TRUE* el sistema retorna el valor "gpBlock_1" el cual es la expresión en R para la función que representa el $Bloque_1$. Si ninguno de los anteriores parámetros no son pasados como argumentos de la función o su valor es *FALSE* entonces la función evaluará el bloque funcional como un modelo $AR(p)$ sin componente de error:

$$gpBlock_1 = \sum_{i=1}^{extParams\$p} params_i * X_{t-i}$$

Dado por la expresión:

$$sum(params[1{:}\,extParams\$p] \ * \ x[1{:}\,extParams\$p])$$

Adicionalmente si existe una incompatibilidad entre los datos de X y la cantidad de parámetros requerida p ó alguno de los valores necesarios para su cálculo están vacíos (null) se retorna un valor de 0.

Lo anterior demuestra la utilidad de R al momento de evaluar expresiones las cuales involucra vectores y matrices como datos nativos del lenguaje, en los cuales se realiza las operaciones aritméticas básicas sin necesidad de utilizar funciones de evaluación, creación y redimensión de vectores adicionales.

Operadores

Los operadores como tal están diseñados para evaluar uno/dos números de entrada de acuerdo con una función que se desea realizando las respectivas validaciones (aunque sea básico el operador se redefine para soportar posibles errores como raíces pares de números negativos, logaritmos negativos, ausencia de argumentos, argumentos no válidos, entre otros). De acuerdo a lo anterior se define su estructura básica debe contener:

- *nombre* corresponde al nombre propio de la función (suele ser nemotécnica, por ejemplo gpSum para el operador aritmético suma $\{+\}$).
- *leftValue* corresponde al primer valor de la función y es obligatorio para cualquier operador.
- *rightValue* corresponde al segundo argumento del operador, dependiendo de la función puede o no ser requerido, por ejemplo para el operador negación (*Not*) no es necesario, pero para el operador suma $\{+\}$ si lo es.
- *snPrint* indica si la función retornará una cadena de caracteres con la representación de la función (equivale en este aspecto al nodo) en forma comprensible para el usuario.
- *snName* indica si la función retornará una cadena de caracteres con el nombre de la función para ser utilizada en una expresión valida en R.

Además de las variables adicionales que sean necesarias para su ejecución por medio '…'. Si los indicadores *snPrint* y *snName* son *FALSE* entonces la función retornará el resultado de ser evaluada para un $leftValue$ y $rightValue$. De acuerdo con lo anterior es posible definir nuevos Operadores que cumplan con la estructura básica antes indicada.

Si alguno de los argumentos es omitido y es necesario para la correcta ejecución, la función retornará 0.

De lo anterior se puede deducir que los operadores son evaluados sobre valores numéricos, que son a su vez resultados de la evaluación de otros operadores y/o terminales. Adicionalmente, si el parámetro *snPrint = TRUE* el sistema retorna el valor "+", el cual es la representación en forma de ecuación (fácilmente

entendible para el usuario) que se definió para el operador. Si el parámetro *snName = TRUE* el sistema retorna el valor "gpSum" el cual es la expresión en R para la función que representa el operador de suma. Si ninguno de los anteriores parámetros no es enviados o su valor es *FALSE*, la función evaluará la suma de los argumentos *leftValue* y *rightValue* enviados de acuerdo con la función definida, para el caso anterior del operador suma, sería equivalente a:

$$gpSum = leftValue + rightValue$$

Dado por la expresión:

$$leftValue + rightValue$$

Lo anterior demuestra la utilidad de R al momento de evaluar operadores a los cuales es necesario realizar verificaciones sobre sus argumentos y la generación del respectivo resultado sin necesidad de utilizar funciones de evaluación, creación y redimensión de variables adicionales.

- *Nodo gpNode*: Constituye la base de generación de los individuos (arboles – ecuaciones) de GP, cada uno de los nodos corresponde a alguno de los tipos:
- *TERMINAL*: Constituye las hojas del árbol, las cuales pueden ser un parámetro (constante) o un bloque funcional, el cual en su forma más básica corresponde a X_{t-i} con $i = 0$.
- *OPERATOR*: Corresponde a los operadores propios que se utilizan para unir y evaluar los terminales, entre ellos podemos encontrar los operadores aritméticos básicos $\{+, -, /, *\}$.

Además, la implementación del nodo es realizada por medio de una lista indexada (dado que la utilización de pilas no es óptimo en un lenguaje no declarativo en el cual el operador es infijo y no pre o post fijo de acuerdo con los análisis realizados por Banzhaf et al. [129]) lo que permite anidar sub-árboles por medio de un nodo padre (nodo inicial), sin necesidad de redimensionar la lista durante cada operación de creación, eliminación y/o actualización de algún nodo; para ello es necesario contar con dos atributos adicionales que permitan almacenar las listas izquierda (*leftNode*) y derecha (*rightNode*) por medio de sus nodos (*gpNodes*) iníciales, los cuales equivalen a los sub-árboles izquierdo y derecho respectivamente.

Dado que los operadores pueden tener 1 ó 2 argumentos (para el caso del SR), es necesario almacenar el atributo nArg que permite realizar las validaciones necesarias para la aplicación de los operadores genéticos, adicional a brindar facilidades en la generación de árboles aleatorios.

Por último se debe tener en cuenta que los nodos retornan valores a partir de la evaluación de una función cuyo resultado más básico es x, por lo que es necesario tener la función a ser evaluada (de acuerdo con los argumentos enviados) y dado que en R es posible tener funciones en variables (puntero a función), se puede agregar un atributo llamado *fun*, la cual contendrá el puntero a la función específica, la cual evalúa el nodo y genera como resultado un valor numérico.

Todas las clases heredan de una genérica llamada *gpGeneralNode* debido a que no es posible definir un atributo y/o variable de un tipo sin antes haber sido creado, y si fuese definido de tipo gpNode se generaría recursividad infinita. A continuación se describen aquellos métodos asociados al *gpNode*:

> gpNodeCountNodes <- function(object = NULL)

Donde object es de tipo *gpNode*. Ésta función retorna el conteo total de nodos del árbol cuyo nodo inicial es object. Si object es *NULL* y/o no posee los atributos básicos de un *gpNode* retornará0.

> gpNodeCountParam <- function(object = NULL, extParams = NULL)
Donde object es de tipo *gpNode* y *extParams* es la lista de parámetros externos de la forma c("nombre parámetro" = valor parámetro). Ésta función retorna el conteo total de parámetros del árbol cuyo nodo inicial es object de acuerdo con la suma de cada uno de los parámetros de los terminales bajo los parámetros externos *extParams*. Si object es *NULL* y/o no posee los atributos básicos de un *gpNode* retornará 0.

> gpNodePrint <- function(object = NULL)
Donde object es de tipo *gpNode*. Ésta función retorna la cadena de caracteres ("character") del árbol cuyo nodo inicial es object de acuerdo con la concatenación de las expresiones de cada una de las funciones de los nodos que componen el árbol al ser evaluadas con argumento *snPrint = TRUE*, en un formato de fácil lectura para las personas. Si object es *NULL* y/o no posee los atributos básicos de un *gpNode* es retornada la cadena vacía "".

> gpNodeEquation <- function(object = NULL, nParam = 1, totalParam = 0)
Donde object es de tipo *gpNode*, *nParam* es la posición desde la cual debe ser tomado los parámetros para el nodo actual de la lista de parámetros *params* y *totalParam* es el número total de elementos de la lista params. Ésta función retorna la cadena de caracteres ("character") del árbol cuyo nodo inicial es *object* de acuerdo con la concatenación de las expresiones de cada una de las funciones de los nodos que componen el árbol (individuo) al ser evaluadas con argumento *snName = TRUE*, en un formato el cual puede ser evaluado como una expresión en R por medio de la función *eval*. Si object es *NULL* y/o no posee los atributos básicos de un *gpNode* retornará la cadena vacía "".

> gpNodeGenRandom <- function(totalNodes = 0, listOperators = NULL, listTerminals = NULL)
Donde *totalNodes* es el número máximo de nodos a ser utilizados en el árbol, *listOperators* es la lista de posibles operadores a ser utilizados (nombre = número de argumentos, donde nombre son los nombres de las funciones que aplican para nodos tipo *OPERATOR*) y *listTerminals* es la lista de posibles terminales a ser utilizadas (nombres de funciones que aplican para nodos tipo *TERMINAL*). Ésta función retorna una lista en la cual se indica el número total de nodos restantes por ser utilizados (*totalNodes*) y el nodo inicial del árbol generado aleatoriamente (*object*). Si *totalNodes <= 0* o *listOperators = NULL* o *listTerminals = 0* se retornará la lista *list("totalNodes" = 0, "object" = new("gpNode")*).

> gpNodeReplace <- function(object = NULL, object2 = NULL, pos = 0)
Donde *object* y *object2* son de tipo gpNode y *pos* es un valor numérico. Ésta función retorna el nodo inicial del árbol formado por object al cual se ha reemplazado el nodo en la posición pos por el árbol object2. Si object es *NULL* o *object2* es *NULL* o *pos <= 1* la función retornará *object*.

> gpNodeSubTree <- function(object = NULL, pos = 0)
Donde *object* es de tipo *gpNode* y *pos* es un valor numérico. Ésta función retorna el nodo inicial del subárbol de *object* tomado a partir de la posición *pos*. Si *object* es *NULL* o *pos <= 1* entonces retornará *object*.

> gpNodeCrossOver <- function(object1 = NULL, object2 = NULL, pos1 = 0, pos2 = 0)
Donde *object1* y *object2* son de tipo *gpNode*, y *pos1* y *pos2* son valores numéricos. Ésta función retorna una lista con los descendientes (*list("object1", "object2")*) del cruce de los arboles *object1* y *object2* en sus respectivas posiciones *pos1* y *pos2* siempre y cuando sea posible dicho cruce. Si *object1* es *NULL* o *object2* es *NULL* o *pos1 <= 1* o *pos2 <= 1* entonces *retornará list("object1" = object1, "object2" = object2*). Esta función corresponde al operador genético de cruce para arboles binarios de GP.

> ***gpNodeMutate <- function(object = NULL, pos = 0, listOperators = NULL, listTerminals = NULL)***
Donde object es de tipo *gpNode*, pos es el valor numérico de la posición del nodo a ser mutado, *listOperators* es la lista de posibles operadores a ser utilizados (nombre = número de argumentos, donde nombre son los nombres de las funciones que aplican para nodos tipo *OPERATOR*) y *listTerminals* es la lista de posibles terminales a ser utilizadas (nombres de funciones que aplican para nodos tipo *TERMINAL*). Ésta función retorna una lista con el nodo inicial del árbol object con el nodo en la posición *pos* mutado equivalente en número de argumentos al existente (*list("object", "pos")*). Si *object* es *NULL* o *pos <= 1* entonces retornará *list("object" = object, "pos" = pos)*. Esta función corresponde al operador genético de mutación para arboles binarios de GP.

> ***gpNodeRestruct <- function(object = NULL, nParam = 1, totalParam = 0, params = NULL, extParams = NULL)***
Donde *object* es de tipo *gpNode*, *nParam* es la posición desde la cual debe ser tomado los parámetros para el nodo actual de la lista de parámetros params y *totalParam* es el número total de elementos de la lista *params* y *extParams* es la lista de parámetros externos. Ésta función retorna una lista *list("object" = object, "nParam" = nParam, "totalParam" = totalParam, "params" = params)* en la que *object* es el nodo inicial del árbol resultante después de aplicar las operaciones de simplificación matemáticas básicas de suma, resta, multiplicación y división entre terminales, *params* es el nuevo vector de parámetros, *nParam* es la posición desde la cual debe ser tomado los parámetros para el nodo resultado y *totalParam* es el total de parámetros cuyo árbol está definido por el nodo inicial *object*.

El operador de reproducción consiste en generar un árbol copia a partir de un nodo inicial *object* y dado que se están utilizando listas y no punteros, basta con asignar a una variable el nodo inicial *object*.

Individuo (Árbol gpTree)
Constituye el individuo tanto en forma de árbol (por medio de una variable de tipo *gpNode* llamada *initalNode*), como expresión de R (variable de tipo texto llamada *funEq*) y su respectivo tamaño (*totalNodes*) y parámetros (*params*). Esta clase se encarga de realizar las operaciones necesarias para dar los resultados a la aplicación sobre individuos de los operadores genéticos.

A continuación se describen aquellos métodos asociados al *gpTree*:

> **gpTreeGenRandom <- function(totalNodes = 0, listOperators = NULL, listTerminals = NULL, minValParam = 0, maxValParam = 1, extParams = NULL)**
Donde *totalNodes* es el número máximo de nodos a ser utilizados en el individuo, *listOperators* es la lista de posibles operadores a ser utilizados (nombre = número de argumentos, donde nombre son los nombres de las funciones que aplican para nodos tipo *OPERATOR*) y *listTerminals* es la lista de posibles terminales a ser utilizadas (nombres de funciones que aplican para nodos tipo *TERMINAL*), *minValParam* es el valor mínimo para ser seleccionado como valor base aleatorio de los parámetros *params* y *maxValParam* es el valor máximo para ser seleccionado como valor base aleatorio de los parámetros *params*, *extParams* es la lista de parámetros externos que aplican para el árbol. Ésta función retorna un individuo de máximo *totalNodes* inicializando la lista de parámetros *params* y contando sus respectivos nodos en *totalNodes* y la expresión valida en R en *funEq*. Si *totalNodes <= 0* o *listOperators = NULL* o *listTerminals = 0* se es retornada la lista *list("totalNodes" = 0, "object" = new("gpNode"))*. Se debe recordar que el valor a ser asignado a cada parámetro params es igual a *runif(gpNodeCountParam(auxNode$object, extParams), minValParam, maxValParam).*

> **gpTreePrint <- function(object = NULL)**
Donde *object* es de tipo *gpTree*. Ésta función retorna la cadena de caracteres ("character") del árbol cuyo nodo inicial es *object@initialNode* de acuerdo con la concatenación de las expresiones de cada una de

las funciones de los nodos que componen el árbol al ser evaluadas con argumento *snPrint = TRUE*, en un formato de fácil lectura para los usuarios. Si *object* es *NULL* y/o no posee los atributos básicos de un *gpTree* es retornada la cadena vacía “”.

> gpTreeReproduction <- function(object = NULL)
Donde *object* es de tipo *gpTree*. Ésta función retorna una copia de *object* y equivale al operador de reproducción entre individuos GP.

> gpTreeCrossOver <- function(object1 = NULL, object2 = NULL, pos1 = 0, pos2 = 0, minValParam = 0, maxValParam = 1, extParams = NULL)
Donde *object1* y *object2* son de tipo *gpTree* y *pos1*, *pos2* son valores numéricos, si son menores que 1 o mayores que el total de nodos por individuo se generan aleatoriamente entre 1 y el total de nodos por individuo, *minValParam* es el valor mínimo para ser seleccionado como valor base aleatorio de los parámetros *params* y *maxValParam* es el valor máximo para ser seleccionado como valor base aleatorio de los parámetros *params* para cada individuo, *extParams* es la lista de parámetros externos que aplican para cada individuo. Ésta función retorna una lista con los descendientes (*list(“object1”, “object2”)*) del cruce de los individuos *object1* y *object2* en sus respectivas posiciones *pos1* y *pos2* siempre y cuando sea posible dicho cruce, actualizando el total de nodos de cada individuo, su expresión *funEq* y la lista de parámetros *params*. Si *object1* es *NULL* o *object2* es *NULL* entonces retornará *list("object1" = object1, "object2" = object2)*. Esta función corresponde al operador genético de cruce para individuos de GP.

> gpTreeMutate <- function(object = NULL, pos = 0, listOperators = NULL, listTerminals = NULL, minValParam = 0,maxValParam = 1, extParams = NULL)
Donde *object* es de tipo *gpTree*, *pos* es el valor numérico de la posición del nodo a ser mutado, *listOperators* es la lista de posibles operadores a ser utilizados (nombre = número de argumentos, donde nombre son los nombres de las funciones que aplican para nodos tipo *OPERATOR*) y *listTerminals* es la lista de posibles terminales a ser utilizadas (nombres de funciones que aplican para nodos tipo *TERMINAL*), *minValParam* es el valor mínimo para ser seleccionado como valor base aleatorio de los parámetros *params* y *maxValParam* es el valor máximo para ser seleccionado como valor base aleatorio de los parámetros params para cada individuo, *extParams* es la lista de parámetros externos que aplican para cada individuo. Ésta función retorna un nuevo individuo de tipo *gpTree* analizando la nueva configuración de la lista de parámetros params. Esta función corresponde al operador genético de mutación para individuos de GP.

> gpTreeRestruct <- function(object = NULL, extParams = NULL)
Donde *object* es de tipo *gpTree* y *extParams* es la lista de parámetros externos que aplican para cada individuo. Ésta función retorna un nuevo individuo de tipo *gpTree* después de ser aplicado el operador *gpNodeRestruct* al nodo inicial initialNode.

> gpTreeAdd <- function(object1 = NULL, object2 = NULL, operFun = NULL, extParams = NULL)
Donde *object1* y *object2* son de tipo *gpTree*, operFun es la función de un operador básico que reciba 2 argumentos (ejemplo el operador suma) y *extParams* es la lista de parámetros externos que aplican para cada individuo. Ésta función retorna el *gpTree object* resultante de generar un nuevo árbol cuyo nodo inicial es de tipo *OPERATOR* con función *operFun* y cuyos sub-árboles izquierdo y derecho corresponden a los *gpTree object1* y *object2* respectivamente.

> ***gpTreeEval <- function(funEq = NULL, x = NULL, params = NULL, extParams = NULL)***
Esta función retorna la evaluación de la expresión en R dada por la cadena *funEq* con base en los valores de *x*, *params* y *extParams*. Es catalogada como función perteneciente a *gpTree* dado que toma que argumento principal el *funEq* de un *gpTree* especifico.

> ***gpTreeOptimizate <- function(funEq = NULL, params = NULL, X = NULL, b = NULL, extParams = NULL, optMethod = NULL, funError = NULL, optOA = NULL, ...)***
Esta función retorna los parámetros optimizados a partir de una lista inicial *params*, evaluado a la luz de la expresión en R dada por la cadena *funEq* para los valores de *x, b*, *extParams* específicos por medio del método *optMethod* (apuntador a función), la función de error especifica *funError* (apuntador a función) y la lista de opciones propias del método de optimización *optOA* (nombre = valor). Es catalogada como función perteneciente a *gpTree* dado que toma que argumento principal el *funEq* de un *gpTree* específico.

De acuerdo a lo anterior un individuo es un árbol cuyo nodo inicial es *initialNode*, la lista de parámetros asociados al árbol corresponden al atributo *params*, el número total de nodos es *totalNodes* y la expresión en R valida a ser evaluada para un x, *params* y *extParams* específicos corresponde a *funEq*.

Modelo gpModel
Constituye el manejador de individuos y el responsable de almacenar la información de opciones y parámetros relativos al algoritmo de GP. A continuación se describen aquellos métodos asociados al *gpModel*:

> ***gpFunIniBasic <- function(maxPop = 0, maxNodes = 0, listOperators = NULL, listTerminals = NULL, minValParam = 0, maxValParam = 1, extParams = NULL, ...)***
Donde *maxPop* es el número máximo de individuos a ser generados, *totalNodes* es el número máximo de nodos a ser utilizados en cada individuo, *listOperators* es la lista de posibles operadores a ser utilizados (nombre = número de argumentos, donde nombre son los nombres de las funciones que aplican para nodos tipo *OPERATOR*) y *listTerminals* es la lista de posibles terminales a ser utilizadas (nombres de funciones que aplican para nodos tipo *TERMINAL*), *minValParam* es el valor mínimo para ser seleccionado como valor base aleatorio de los parámetros *params* y *maxValParam* es el valor máximo para ser seleccionado como valor base aleatorio de los parámetros *params*, *extParams* es la lista de parámetros externos que aplican para el árbol. Ésta función retorna un vector con los individuos de la población inicial con un máximo de nodos dado por *totalNodes* inicializando la lista de parámetros *params* y contando sus respectivos nodos en *totalNodes* y la expresión valida en R en *funEq*. Si *totalNodes <= 0* o *listOperators = NULL* o *listTerminals = 0* se retornará la lista *c(new("gpTree"))*. Se debe recordar que el valor a ser asignado a cada parámetro params es igual a *runif(gpNodeCountParam(auxNode$object, extParams), minValParam, maxValParam)*.

> ***gpModelGPAlgBasic <- function(object = NULL, ...)***
Donde object es de tipo *gpModel*. Ésta función retorna un objeto de tipo *gpModel* con el resultado de la ejecución del algoritmo de GP para el modelo *object* actualizando los atributos de *bestInd*, *htcoAvgInd*, *htcoMaxInd*, *htcoMinInd* y *actError* de acuerdo con su función de parada *funEnd*. Si *object* es *NULL* o *object@X* es *NULL* o *object@b* es *NULL* o *object@maxPop* es *NULL* o *object@maxPop < 1* o *object@maxGen* es *NULL* o *object@maxGen < 1* o *object@maxNodes* es *NULL* o *object@maxNodes < 1* retorna *object*.

Dada la definición(es) del algoritmo de GP, éste es susceptible a ser optimizado por cualquier método de optimización (como cualquier modelo de regresión) que sea implementado en la estructura exigida por el

modelo, de acuerdo a ello, es posible definir nuevos métodos de optimización que cumplan con el siguiente formato de argumentos:

> ***nombre_funcion <- function(funEq = NULL, X = NULL, b = NULL, params = NULL, extParams = NULL, optGA = NULL, ...)***
Donde *nombre_funcion* es el nombre de la función específica a ser utilizada (se recomienda que sea nemotécnica y con prefijo gp), *funEq* es la cadena que representa una expresión valida en R a ser evaluada con $x = X_{t-i}$, con función de aptitud constatada contra el vector de salida *b*, a partir de los parámetros *params*, *extParams* y demás dados por '…'.

De acuerdo con lo anterior, la estructura de la implementación en R le permite al usuario/analista definir funciones adicionales de evaluación de operadores, terminales, optimización, evaluación del error, criterios de parada e inicialización de la población, de acuerdo con las necesidades y características propias del fenómeno a analizar, lo anterior permite la introducción de conocimiento experto a priori al modelo, lo que permite limitar el espacio de búsqueda y el uso de una gama más amplia de funciones.

Funciones generales
Para el adecuado desarrollo de los métodos inmersos en el algoritmo de GP son necesarias una serie de funciones adicionales las cuales se describen a continuación:

> ***gpFunOptOPTIM <- function(funEq = NULL, X = NULL, b = NULL, params = NULL, extParams = NULL, optEE = NULL, ...)***
Esta función retorna la optimización de params por medio del algoritmo de OPTIM (General-purpose Optimization) [109] sobre la expresión en R dada por la cadena *funEq* para los valores *X, b, params, extParams*, las opciones propias del algoritmo *optBP* y las opciones adicionales dadas por '…'. Se pueden definir otras funciones de optimización respetando el estándar de argumentos y retorno.

> ***gpFunOptGenoudOPT<- function(funEq = NULL, X = NULL, b = NULL, params = NULL, extParams = NULL, optEE = NULL, ...)***
Esta función retorna la optimización de *params* por medio del algoritmo de GENOUD (Genetic Optimization Using Derivatives) sobre la expresión en R dada por la cadena *funEq* para los valores *X, b, params, extParams*, las opciones propias del algoritmo *optBP* y las opciones adicionales dadas por '…'. Se pueden definir otras funciones de optimización respetando el estándar de argumentos y retorno.

> ***gpFunSelTournament <- function(profit = NULL, selCrossSubN = NULL,...)***
Esta función retorna el índice del individuo a ser seleccionada para la aplicación del operador de Cruce por medio del algoritmo de Torneo a partir del vector de *profit* y el número máximo de sub individuos *selCrossSubN* a ser evaluados.

> ***gpFunSelFUSS <- function(profit = NULL, ...)***
Esta función retorna el índice del individuo a ser seleccionada para la aplicación del operador de Cruce por medio del algoritmo de selección proporcional a la función de aptitud partir del vector de *profit*.

> ***gpErrorSSE <- function(xVal = NULL, b = NULL, ...)***
Esta función retorna el error entre los vectores *xVal* y *b* como la suma de sus diferencias componente a componente (dimensiones) al cuadrado: *sum((xVal - b) ^ 2)*. Si *xVal = NULL* o *b = NULL* entonces retornará -1.

> gpErrorMSE <- function(xVal = NULL, b = NULL, ...)

Esta función retorna el error entre los vectores *xVal* y *b* como la suma promedio de sus diferencias componente a componente (dimensiones) en valor absoluto*: sum(abs(xVal - b)) / dim(b)[1]*. Si *xVal = NULL* o *n = NULL* entonces retornará −1.

> gpErrorAkaike <- function(xVal = NULL, b = NULL, params = NULL, ...)

Esta función retorna el error entre los vectores *xVal* y *b* aplicando el criterio Akaike [117] como la expresión *log(sum((xVal - b)^2) / dim(b)[1]) + ((dim(b)[1] + length(params))/(dim(b)[1] - length(params) - 2)*. Si *xVal = NULL* o *b = NULL* o *params = NULL* entonces retornará −1.

> gpFunEndBasic <- function(actError = 0, befError = 0, epsilon = 0, ...)

Esta función retorna *TRUE* si el error actual *actError* es 0 o la diferencia entre el error actual y el error anterior *befError* en valor absoluto es menor o igual a *(epsilon: actError == 0 || abs(actError - befError) <= epsilon)*.

Bibliografía

[1] J. G. De Gooijer and R. J. Hyndman, "25 years of time series forecasting," *International Journal of Forecasting*, vol. 22, no. 3, pp. 443–473, 2006.

[2] N. K. Kasabov, "Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering," *The MIT Press*, 1996.

[3] J. R. Koza, "Genetic Programming: On the Programming of Computers by Means of Natural Selection," *Cambridge: MIT Press*, 1992.

[4] M. A. Kaboudan, "Genetic programming prediction of stock prices," *Computational Economics*, vol. 16, no. 3, pp. 207–236, 2000.

[5] W. C. Wang, K. W. Chau, C. T. Cheng, and L. Qiu, "A comparison of performance of several artificial intelligence methods for forecasting monthly discharge time series," *Journal of Hydrology*, vol. 374, no. 3–4, pp. 294–306, 2009.

[6] W. Abdelmalek, S. B. Hamida, and F. Abid, "Selecting the best forecasting-implied volatility model using genetic programming," *Journal of Applied Mathematics and Decision Sciences*, vol. 2009, 2009.

[7] N. Y. Nikolaev and H. Iba, "Regularization approach to inductive genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 4, pp. 359–375, 2001.

[8] T. H. Hoang, R. I. McKay, D. Essam, and X. H. Nguyen, "Solving symbolic regression problems using incremental evaluation in genetic programming," in *2006 IEEE Congress on Evolutionary Computation, CEC 2006*, pp. 2134–2141, 2006.

[9] C. E. Borges, C. L. Alonso, and J. L. Montaña, "Model selection in genetic programming," in *Proceedings of the 12th Annual Genetic and Evolutionary Computation Conference, GECCO '10*, pp. 985–986, 2010.

[10] M. Johnston, T. Liddle, and M. Zhang, "A relaxed approach to simplification in genetic programming," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6021 LNCS. School of Mathematics, Statistics and Operations Research, P.O. Box 600, Wellington, New Zealand, pp. 110–121, 2010.

[11] S. Ok, K. Miyashita, and S. Nishihara, "Improving performance of GP by adaptive terminal selection," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1886 LNAI. Electrotechnical Laboratory (ETL), 1-1-4, Umezono, Tsukuba, Ibaraki, 305-8568, Japan, pp. 435–445, 2000.

[12] C. A. Martinez, *"*Problemas abiertos en la aplicación de la Regresión Simbólica en el pronóstico de series de tiempo,*" Tesis maestría, Universidad Nacional de Colombia sede Medellín*, 2011.

[13] G. Box, G. Jenkins, "Time series analysis: Forecasting and control," *San Francisco: Holden-Day*, 1970.

[14] S. Sette, L. Boullart, "Genetic Programming: Principles and applications", *Engineering Aplications of Artificial Intelligence*, vol. 14, pp. 727 – 736, 2001.

[15] C. Genolini, "A (Not So) Short Introduction to S4," *Object Oriented Programming in R V0.5.1*, 2008.

[16] M. Ghiassi, H. Saidane and D. K. Zimbra, "A dynamic artificial neural network model for forecasting time series events," *International Journal of Forecasting*, vol. 21, pp. 341– 362, 2005.

[17] J. D. Velásquez, Y. Olaya and C. J. Franco, "Predicción de series temporales usando máquinas de vectores de soporte,*" Ingeniare. Revista chilena de ingeniería*, v. 18 n.1, pp. 64-75, 2010.

[18] B. A. Kitchenham, T. Dybå, and M. Jørgensen, "Evidence-based software engineering," in *Proceedings - International Conference on Software Engineering*, 2004, vol. 26, pp. 273–281.

[19] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, "Lessons from applying the systematic literature review process within the software engineering domain," *Journal of Systems and Software*, vol. 80, no. 4, pp. 571–583, 2007.

[20] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering - A systematic literature review," *Information and Software Technology*, vol. 51, no. 1, pp. 7–15, 2009.

[21] F. Vafaee, P. C. Nelson, C. Zhou, and W. Xiao, "Dynamic adaptation of genetic operators' probabilities," *Studies in Computational Intelligence*, vol. 129. Artificial Intelligence Laboratory, University of Illinois at Chicago, Chicago, IL 60607, United States, pp. 159–168, 2008.

[22] A. Hara, M. Watanabe, and T. Takahama, "Cartesian ant programming," in *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, pp. 3161–3166, 2011.

[23] L. Zhao, L. Wang, and D. W. Cui, "Hoeffding bound based evolutionary algorithm for symbolic regression," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 5, pp. 945–957, 2012.

[24] N. F. McPhee and R. Poli, "Memory with memory: Soft assignment in genetic programming," in *GECCO'08: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation 2008*, pp. 1235–1242, 2008.

[25] D. T. Xie, L. S. Kang, Y. Q. Li, and X. Du, "Novel algorithm for symbolic regression," *Xitong Fangzhen Xuebao / Journal of System Simulation*, vol. 19, no. 8, pp. 1667–1671, 2007.

[26] S. W. Xiong and W. W. Wang, "Point-tree structure genetic programming method for discontinuous function's regression," *Wuhan University Journal of Natural Sciences*, vol. 8, no. 1 B, pp. 323–326, 2003.

[27] K. Ono, Y. Hanada, K. Shirakawa, M. Kumano, and M. Kimura, "Depth-dependent crossover in genetic programming with frequent trees," in *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, pp. 359–363, 2012.

[28] Y. Hanada, N. Hosokawa, K. Ono, and M. Muneyasu, "Effectiveness of multi-step crossover fusions in genetic programming," in *2012 IEEE Congress on Evolutionary Computation, CEC 2012*, 2012.

[29] E. Semenkin and M. Semenkina, “Self-configuring genetic programming algorithm with modified uniform crossover,” in *2012 IEEE Congress on Evolutionary Computation, CEC 2012*, 2012.

[30] N. Q. Uy, M. O’Neill, N. X. Hoai, B. Mckay, and E. Galván-López, “Semantic similarity based crossover in GP: The case for real-valued function regression,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5975 LNCS. Natural Computing Research and Applications Group, University College Dublin, Ireland, pp. 170–181, 2010.

[31] L. Vanneschi and S. Gustafson, “Using crossover based similarity measure to improve genetic programming generalization ability,” in *Proceedings of the 11th Annual Genetic and Evolutionary Computation Conference, GECCO-2009*, pp. 1139–1146, 2009.

[32] M. D. Terrio and M. I. Heywood, “Directing crossover for reduction of bloat in GP,” in *Canadian Conference on Electrical and Computer Engineering*, vol. 2, pp. 1111–1115, 2002.

[33] A. Agapitos, A. Brabazon, and M. O’Neill, “Controlling overfitting in symbolic regression based on a bias/variance error decomposition,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7491 LNCS, no. PART 1. Financial Mathematics and Computation Research Cluster, Natural Computing Research and Applications Group, University College Dublin, Ireland, pp. 438–447, 2012.

[34] J. L. Montaña, C. L. Alonso, C. E. Borges, and J. De La Dehesa, “Penalty functions for genetic programming algorithms,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6782 LNCS, no. PART 1. Departamento de Matemáticas, Estadística Y Computación, Universidad de Cantabria, 39005 Santander, Spain, pp. 550–562, 2011.

[35] J. H. Imada and B. J. Ross, “Using feature-based fitness evaluation in symbolic regression with added noise,” in *GECCO’08: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation 2008*, pp. 2153–2157, 2008.

[36] L. Zhang and A. K. Nandi, “Neutral offspring controlling operators in genetic programming,” *Pattern Recognition*, vol. 40, no. 10, pp. 2696–2705, 2007.

[37] H. Xie, M. Zhang, and P. Andreae, “Population clustering in genetic programming,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, LNCS. School of Mathematics Statistics and Computer Science, Victoria University of Wellington, P.O. Box 600, Wellington, New Zealand, vol. 3905, pp. 190–201, 2006.

[38] M. Tomassini, L. Vanneschi, J. Cuendet, and F. Fernández, “A new technique for dynamic size populations in genetic programming,” in *Proceedings of the 2004 Congress on Evolutionary Computation, CEC2004*, vol. 1, pp. 486–493, 2004.

[39] L. Sánchez, “Interval-valued GA-P algorithms,” *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 1, pp. 64–72, 2000.

[40] Karaboga, C. Ozturk, N. Karaboga, and B. Gorkemli, "Artificial bee colony programming for symbolic regression," *Information Sciences*, vol. 209, pp. 1–15, 2012.

[41] K. Y. Chan and C. K. Kwong, "Modeling of epoxy dispensing process using a hybrid fuzzy regression approach," *International Journal of Advanced Manufacturing Technology*, vol. 65, issue 1-4, pp 589-600, 2013.

[42] G. Kronberger and A. Beham, "Symbolic regression using tabu search in a neighborhood of semantically similar solutions," in *24th European Modeling and Simulation Symposium, EMSS 2012*, pp. 379–384, 2012.

[43] P. F. Guo, P. Bhattacharya, and N. Kharma, "Automated synthesis of feature functions for pattern detection," in *Canadian Conference on Electrical and Computer Engineering*, 2010.

[44] C. E. Borges, C. L. Alonso, J. L. Montaña, M. De La Cruz Echeandía, and A. O. De La Puente, "Coevolutionary architectures with straight line programs for solving the symbolic regression problem," in *ICEC 2010 - Proceedings of the International Conference on Evolutionary Computation*, pp. 41–50, 2010.

[45] A. H. Gandomi, A. H. Alavi, P. Arjmandi, A. Aghaeifar, and R. Seyednour, "Genetic programming and orthogonal least squares: A hybrid approach to modeling the compressive strength of CFRP-confined concrete cylinders," *Journal of Mechanics of Materials and Structures*, vol. 5, no. 5, pp. 735–753, 2010.

[46] C. L. Pennachin, M. Looks, and J. A. De Vasconcelos, "Robust symbolic regression with affine arithmetic," in *Proceedings of the 12th Annual Genetic and Evolutionary Computation Conference, GECCO '10*, pp. 9–14, 2010.

[47] S. Y. Yuen and S. W. Leung, "Genetic programming that ensures programs are original," in *2009 IEEE Congress on Evolutionary Computation, CEC 2009*, pp. 860–866, 2009.

[48] A. J. Barton, J. J. Valdés, and R. Orchard, "Neural networks with multiple general neuron models: A hybrid computational intelligence approach using Genetic Programming," *Neural Networks*, vol. 22, no. 5–6, pp. 614–622, 2009.

[49] C. Fillon and A. Bartoli, "Symbolic regression of discontinuous and multivariate functions by Hyper-Volume Error Separation (HVES)," in *2007 IEEE Congress on Evolutionary Computation, CEC 2007*, pp. 23–30, 2007.

[50] O. Giustolisi and D. A. Savic, "A symbolic data-driven technique based on evolutionary polynomial regression," *Journal of Hydroinformatics*, vol. 8, no. 3, pp. 207–222, 2006.

[51] L. M. de Menezes and N. Y. Nikolaev, "Forecasting with genetically programmed polynomial neural networks," *International Journal of Forecasting*, vol. 22, no. 2, pp. 249–265, 2006.

[52] G. Smits and E. Vladislavleva, "Ordinal Pareto genetic programming," in *2006 IEEE Congress on Evolutionary Computation, CEC 2006*, pp. 3114–3120, 2006.

[53] O. Costa and A. Pozo, "A $(\mu + \lambda)$ - GP algorithm and its use for regression problems," in *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, pp. 10–17, 2006.

[54] S. Cagnoni, D. Rivero, and L. Vanneschi, “A purely evolutionary memetic algorithm as a first step towards symbiotic coevolution,” in *2005 IEEE Congress on Evolutionary Computation, IEEE CEC 2005. Proceedings*, vol. 2, pp. 1156–1163, 2005.

[55] S. Jiang, Z. Cai, D. Zeng, Y. Liu, and Q. Li, “Gene expression programming based on simulated annealing,” in *Proceedings - 2005 International Conference on Wireless Communications, Networking and Mobile Computing, WCNM 2005*, vol. 2, pp. 1218–1221, 2005.

[56] S. W. Jiang, Z. H. Cai, D. Zeng, Q. Li, and Y. F. Cheng, “Parallel gene expression programming algorithm based on simulated annealing method,” *Tien Tzu Hsueh Pao/Acta Electronica Sinica*, vol. 33, no. 11, pp. 2017–2021, 2005.

[57] M. Oltean, “Solving even-parity problems using traceless genetic programming,” in *Proceedings of the 2004 Congress on Evolutionary Computation, CEC2004*, vol. 2, pp. 1813–1819, 2004.

[58] G. Y. Lee, “Time series perturbation by genetic programming,” in *Proceedings of the IEEE Conference on Evolutionary Computation, ICEC*, vol. 1, pp. 403–409, 2001.

[59] L. M. Howard and D. J. D’Angelo, “GA-P: a genetic algorithm and genetic programming hybrid,” *IEEE expert*, vol. 10, no. 3, pp. 11–15, 1995.

[60] A. Zăvoianu, G. Kronberger, M. Kommenda, D. Zaharie and M. Affenzeller, “Improving the Parsimony of Regression Models for an Enhanced Genetic Programming Process,” *Lecture Notes in Computer Science*, vol. 6927, pp 264-271, 2012.

[61] M. Tanji and H. Iba, “A new GP recombination method using random tree sampling,” *IEEJ Transactions on Electronics, Information and Systems*, vol. 130, no. 5, pp. 775–781, 2010.

[62] N. Mori, B. McKay, N. X. Hoai, D. Essam, and S. Takeuchi, “A new method for simplifying algebraic expressions in genetic programming called equivalent decision simplification,” *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 13, no. 3, pp. 237–244, 2009.

[63] D. Kinzett, M. Johnston, and M. Zhang, “How online simplification affects building blocks in genetic programming,” in *Proceedings of the 11th Annual Genetic and Evolutionary Computation Conference, GECCO-2009*, pp. 979–986, 2009.

[64] M. Zhang and P. Wong, “Explicitly simplifying evolved genetic programs during evolution,” *International Journal of Computational Intelligence and Applications*, vol. 7, no. 2, pp. 201–232, 2008.

[65] Y. Chen, C. Tang, C. Li, Y. Wang, N. Yang, and M. Zhu, “HDN-GEP: A novel gene expression programming with high density node,” in *Proceedings - International Conference on Intelligent Computation Technology and Automation, ICICTA 2008*, vol. 1, pp. 60–64, 2008.

[66] B. M. Cerny, P. C. Nelson, and C. Zhou, “Using differential evolution for symbolic regression and numerical constant creation,” in *GECCO’08: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation 2008*, pp. 1195–1202, 2008.

[67] O. Muntean, L. Diosan, and M. Oltean, “Best SubTree genetic programming,” in *Proceedings of GECCO 2007: Genetic and Evolutionary Computation Conference*, pp. 1667–1673, 2007.

[68] X. Li, C. Zhou, W. Xiao, and P. C. Nelson, "Introducing emergent loose modules into the learning process of a linear genetic programming system," in *Proceedings - 5th International Conference on Machine Learning and Applications, ICMLA 2006*, pp. 219–224, 2006.

[69] M. Aichour and M. Batouche, "A selective mutation operator for tree-based genetic programming," *International Review on Computers and Software*, vol. 4, no. 1, pp. 101–106, 2009.

[70] Y.-Y. Tao, J. Cao, and M.-L. Li, "Genetic programming using dynamic population variation for computational efforts reduction in system modeling," *Journal of Shanghai Jiaotong University (Science)*, vol. 17, no. 2, pp. 190–196, 2012.

[71] J. McDermott, U.-M. O'Reilly, L. Vanneschi, and K. Veeramachaneni, "How far is it from here to there? A distance that is coherent with GP operators," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, EvoDesignOpt, CSAIL, MIT, United States, vol. 6621, pp. 190–202, 2011.

[72] G. K. Kronberger, S. M. Winkler, M. Affenzeller, M. Kommenda, and S. Wagner, "Effects of mutation before and after offspring selection in genetic programming for symbolic regression," in *22th European Modeling and Simulation Symposium, EMSS 2010*, pp. 37–42, 2010.

[73] R. McRee, "Symbolic regression using nearest neighbor indexing," in *Proceedings of the 12th Annual Genetic and Evolutionary Computation Conference, GECCO '10 - Companion Publication*, pp. 1983–1989, 2010.

[74] L. Vanneschi and G. Cuccu, "A study of genetic programming variable population size for dynamic optimization problems," in *IJCCI 2009 - International Joint Conference on Computational Intelligence, Proceedings*, pp. 119–126, 2009.

[75] G. Wilson and M. Heywood, "Probabilistic Adaptive Mapping Developmental Genetic Programming (PAM DGP): A new developmental approach," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Faculty of Computer Science, Dalhousie University, 6050 University Ave., Halifax, NS B3H 1W5, Canada, vol. 4193, pp. 751–760, 2006.

[76] H. Xie, "Diversity control in GP with ADF for regression tasks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, School of Mathematics, Statistics, and Computer Science, Victoria University of Wellington, P. O. Box 600, Wellington, New Zealand, vol. 3809, pp. 1253–1257, 2005.

[77] D. Rochat, M. Tomassini, and L. Vanneschi, "Dynamic size populations in distributed genetic programming," in *Lecture Notes in Computer Science*, vol. 3447, pp. 50–61, 2005.

[78] M. De Arruda P., C. A. Davis J, E. Gontijo C. and J. A. de Vasconcelos, *"*A niching genetic programming-based multi-objective algorithm for hybrid data classification,*" Neurocomputing*, vol. 133, pp. 342-357, 2014.

[79] B. Zamani, A. Akbari and B. Nasersharif, "Evolutionary combination of kernels for nonlinear feature transformation,*" Information Science*, vol. 274, pp. 95-107, 2014.

[80] M. Amir H, M. Mehdi E. and G. Folino, "Improving GP generalization: a variance-based layered learning approach," *Genetic Programming and Evolvable Machines*, 2014.

[81] W. Banzhaf, P. Nordin, R. E. Keller, and Frank D. Francone, "Genetic Programming: An introduction," *San Fransisco, CA: Morgan Kaufmann*, 1998.

[82] B. Zhang and H. Muhlenbein, "Adaptive Fitness Functions for Dynamic Growing/Pruning of Program Trees," *Advances in Genetic Programming*, vol. 2, pp. 241-255, 1996.

[83] Y. S. Lee and L. I. Tong, "Forecasting time series using a methodology based on autoregressive integrated moving average and genetic programming," *Knowledge-Based Systems*, vol. 24, no. 1, pp. 66-72, 2011.

[84] I. J. Leontaritis and S. A. Billings, "Input-output parametric models for non-linear systems," *Int. Journal of Control*, vol. 41, no. 2, pp. 303-344, 1985.

[85] J. D. Hamilton, "Time Series Analysis," *Princeton University Press*, Princeton, 1994.

[86] Tong, "Non-Linear Time Series: A Dynamical System Approach," *Oxford: Oxford University Press*, 1990.

[87] J. D. Petruccelli and S. W. Woolford, "A threshold AR(1) model," *J. Appl. Probab*, vol. 21, pp. 270-286, 1984

[88] R. S. Tsay, "Testing and modeling threshold autoregressive processes," *Journal of the American Statistical Association*, vol. 84, no. 405, pp. 231-240, 1989.

[89] K. S. Chan, "Consistency and Limiting Distribution of the Least Squares Estimator of a Threshold Autoregressive Model," *The Annals of Statistics*, vol. 21, pp. 520-533, 1993.

[90] K. S. Chan and S. Tsay, "Limiting Properties of the Least Squares Estimator of a Continuous Threshold Autoregressive Model," *Biometrika*, vol. 45, pp. 413-426, 1998.

[91] B. E. Hansen, "Inference in TAR Models," *Studies in Nonlinear Dynamics and Econometrics*, vol. 1, pp. 119-131, 1997.

[92] Tsay, "Testing and Modeling Multivariate Threshold Models," *J. of American Statistical Assn*, vol. 93, no. 443, pp. 1188-1202, 1998.

[93] A. L. Montgomery, V. Zarnowitz, R. S. TSAY and G. C. Tiao, "Forecasting the U.S. Unemployment Rate," *Journal of the American Statistical Association*, vol. 93, pp. 478-493, 1998.

[94] R. F. Engle, "Autoregressive Conditional Heteroskedasticity with Estimates of the Variance of U.K. Inflation," *Econometrica*, vol. 50, pp. 987 – 1008, 1982.

[95] T. Bollerslev, R. F. Engle and D. B. Nelson, "ARCH Models," *R.F. Engle & D.L. McFadden(eds) Handbook of Econometrics*, vol. IV, Elsevier Science B.V., 1994.

[96] J. L.Kling and D. Bessler, "Forecasting vector autoregressions with Bayesian priors," *American Journal of Agricultural Economics*, vol. 68, pp. 144-151, 1985.

[97] R. Chen and S. Tsay, "Functional-coefficient autoregressives models," *J. Amer. Statist. Assoc.* vol. 88, pp. 298-308, 1993.

[98] V. Haggan and T. Ozaki, "Modeling nonlinear vibrations using an amplitude-dependent autoregressive time series model," *Biometrika*, vol. 68, 189-96, 1981.

[99] Z. Cai, J. Fan, and Q. Yao, "Functional-coefficient regression models for nonlinear time series," *Journal of the American Statistical Association*, vol. 95, n. 451, pp. 941–956, 2000.

[100] C.W.J. Granger and A.P. Andersen, "On the invertibility of time series modelsStochastic Process," *Appl.*, vol. 8, pp. 87–92, 1978.

[101] J. Liu and P. J. Brockwell, "On the general bilinear time series model," *J. Appl. Probab*, v. 18, 617-627, 1988.

[102] C. W. J. Grander and A. P. Andersen, "An Introduction to Bilinear Time Series Models," *Vandenhoeck and Ruprecht, Gettingen*, 1978.

[103] K. S. Chan and H. Tong, "On the Use of the Deterministic Lyapunov Function for the Ergodicity of Stochastic Difference Equations," *Advances in Applied Probability*, vol. 17, pp. 667-678, 1985.

[104] T. Teräsvirta, "Specification, estimation, and evaluation of smooth transition autoregressive models," *Journal of the American Statistical Association*, vol. 89, pp. 208–218, 1994.

[105] V. D. Dick, T. Teräsvirta and H. B. F. Franses, "Smooth transition autoregressive models - A survey of recent developments," *Econometric Reviews, Taylor and Francis Journals*, vol. 21, no. 1, pp. 1-47, 2002.

[106] J. L. Aznarte, J. M. Benítez and J. L. Castro, "Smooth transition autoregressive models and fuzzy rule-based systems: Functional equivalence and consequences," *Fuzzy Sets and Systems* vol. 158, pp. 2734–2745, 2007.

[107] H. White, "Some asymptotic results for learning in single hidden layer feedforward network models," *Journal of the American Statistical Association*, vol. 84, pp. 1003–1013, 1989.

[108] C. W. Granger, T. Teräsvirta and H. M. Anderson, "Modeling Nonlinearity over the Business Cycle," *ch. 8 in J. H. Stock and M. W. Watson (eds.), Business Cycles, Indicators, and Forecasting, Chicago: University of Chicago Press for NBER*, pp. 311-27, 1993.

[109] C. J. P. Belisle, "Convergence theorems for a class of simulated annealing algorithms on R," *J Applied Probability*, vol. 29, pp. 885-895, 1992.

[110] A. Nielsen and H. Madsen, "A generalization of some classical time series tools," *Computational Statistics & Data Analysis*, 2000.

[111] C. A. Martinez, "Análisis de dependencias no lineales utilizando redes neuronales artificiales," *Tesis pregrado, Universidad Nacional de Colombia sede Medellín*, 2006.

[112] H. B. Mann, "Nonparametric tests against trend," *Econometrica*, vol. 13, pp. 245-259, 1945.

[113] M. G. Kendall, "Rank Correlation Methods," *Charles Griffin: London*, 1975.

[114] D. A. Dickey and W. A. Fuller, "Distribution of the Estimators for Autoregressive Time Series with a Unit Root," *Journal of the American Statistical Association*, vol. 74, pp. 427–431, 1979.

[115] M. B. Priestley, "Spectral Analysis and Time Series," *New York, NY: Academic Press*, 1981.

[116] F. Canoa, "Three tests for the existence of cycles in time series," *Ricerche Economiche*, vol. 50, pp.135–162, 1996.

[117] H. Akaike, "Statistical predictor identification," *Ann. Inst. Statist. Math*, vol. 22, pp. 203-17, 1970.

[118] J. Bernardo and A. Smith, "Bayesian theory," *Jhon Wiley*, 1994.

[119] V. Vapnik, "Statistical Learning Theory," *Jhon Wiley*, 1998.

[120] W. R. Mebane and J. S. Sekhon, "Genetic Optimization Using Derivatives: The rgenoud Package for R", *Journal of Statistical Software*, vol. 42, issue 11, 2011.

[121] C. C. Ribeiro and I. Rosseti, "*tttplots-compare: A perl program to compare time-to-target plots or general runtime distributions of randomized algorithms*", Optimization Letters, Springer Berlin Heidelberg, pp. 601-614, 2014.

[122] F. Wilcoxon, "Individual Comparisons by Ranking Methods," *Biometrics Bulletin*, vol. 1, n. 6, pp. 80-83, 1945.

[123] J. Faraway and C. Chatfield, "Time series forecasting with neural networks: a comparative study using the air line data," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 47, issue 2, pp. 231–250, 1998.

[124] M. J. Campbell and A. M. Walker, "A survey of statistical work on the McKenzie River series of annual Canadian lynx trappings for the years 1821–1934, and a new analysis," *J. Roy. Statist. Soc. A*, vol. 140, pp. 411-431, 1977.

[125] G. Zhang, "Time Series forecasting using a hybrid ARIMA and neural network model," *Neurocomputing*, vol. 50, pp. 159–175, 2003.

[126] S. G. Makridakis, S. C. Wheelwright and R. J. Hyndman, "Forecasting: Methods and applications," *John Wiley & Sons*, 3rd edition, New York, USA, ISBN 978-0471532330, 1998.

[127] M. Cottrell, B. Girard and P. Rousset, "Long term forecasting by combining kohonen algorithm and standard prevision," *Lecture notes in computer science*, vol. 1327, pp. 993-998, 1997.

[128] J. M. Chambers and T. Hastie, "Statistical Models in S," *Wadsworth & Brooks/Cole*, 1992.

[129] W. Banzhaf, P. Nordin, R. E. Keller and F. D. Francone, "Genetic programming: an introduction: on the automatic evolution of computer programs and its applications," *Morgan Kaufmann Publishers Inc*., San Francisco, CA, USA, 1998.