UNIVERSIDAD NACIONAL DEL ESTE

FACULTAD POLITÉCNICA

MAESTRÍA EN INFORMÁTICA Y COMPUTACIÓN

METODOLOGÍA PARA LA PREDICCIÓN DE INGRESOS DE CAUSAS PENALES MEDIANTE PROGRAMACIÓN GENÉTICA LINEAL

Orientador: D.Sc. Benjamín Barán Cegla

TESIS DE MAESTRÍA

Alberto David Garcete Rodríguez

2014
Ciudad del Este – Paraguay

Alberto David Garcete Rodríguez

METODOLOGÍA PARA LA PREDICCIÓN DE INGRESOS DE CAUSAS PENALES MEDIANTE PROGRAMACIÓN GENÉTICA LINEAL

Tesis presentada a la Universidad Nacional del Este como requisito parcial para la obtención del título de Magister en Informática y Computación

Orientador: D.Sc. Benjamín Barán Cegla

2014
Ciudad del Este – Paraguay

Garcete Rodríguez, A. D. (2014). Metodología para la Predicción de Ingresos de Causas Penales Mediante Programación Genética Lineal.
Alberto David Garcete Rodríguez 113 páginas.
Orientador: D.Sc. Benjamín Barán Cegla.

Tesis Académica de Maestría en Ciencias Aplicadas.
Universidad Nacional del Este, 2014.

Alberto David Garcete Rodríguez

METODOLOGÍA PARA LA PREDICCIÓN DE INGRESOS DE CAUSAS PENALES MEDIANTE PROGRAMACIÓN GENÉTICA LINEAL

Esta tesis fue evaluada y aprobada para la obtención del título de Magister en Informática y Computación por la Universidad Nacional del Este.

Mesa Examinadora

DEDICATORIA

Este trabajo dedico especialmente a mis queridos padres Isaac y Agueda porque siempre serán mis mejores maestros de la vida.

A mis hermanos de sangre y alma Lidio Jesús, María Visitación, María Concepción, Violeta María Isabel, por la lucha diaria que cada uno enfrenta en este mundo. Y a mí Ángel Félix Javier porque siempre está ahí para cuidar y guiar en este camino.

Y a los verdaderos amigos, que siempre estuvieron en el momento indicado para darme el aliento necesario para afrontar este largo trecho.

AGRADECIMIENTOS

Agradezco a una gran persona, profesor, guía, orientador y un profesional intachable, el D.Sc. Ing. Benjamín Barán Cegla, uno por haberme cedido parte de su conocimiento en la materia abordada durante la Maestría y dos por haberme aceptado como pupilo en la elaboración de esta tesis, donde me ha orientado con su vasto conocimiento.

A la Lic. Lidia Benítez de Pérez, quien fue la propulsora del estímulo para mi inscripción a esta Maestría, invitándome personalmente y recibiéndome con brazos abiertos en la Institución del cual fui egresado.

Al Decano de la Facultad Politécnica Ing. M.Sc. Eustaquio Alcides Martínez, por su apoyo incondicional en esta etapa final del proceso.

Al Señor Coordinador, Dr. David La Red Martínez y su excelente plantel de Docentes, quienes fueron los artífices principales durante estos años de esfuerzo y dedicación para lograr la culminación exitosa de la Maestría en Informática y Computación.

Por último, a mis compañeros de estudio, en especial a Sandra Jiménez, Mirta Arrua y Sergio Morel, con quienes compartí interminables horas de estudio en cada etapa de esta lucha que llega a su fin.

MUCHÍSIMAS GRACIAS A TODOS

Y QUE DIOS LOS BENDIGA

ÍNDICE DE CONTENIDOS

ÍNDICE DE CONTENIDOS ..... **I**

ÍNDICE DE FIGURAS ..... **IV**

ÍNDICE DE TABLAS ..... **V**

LISTA DE ACRÓNIMOS ..... **VI**

LISTA DE SÍMBOLOS ..... **VII**

RESUMEN ..... - 1 -

ABSTRACT ..... - 2 -

1 CAPITULO I INTRODUCCIÓN ..... **- 3 -**

1.1 INTRODUCCIÓN ..... - 4 -

1.2 HIPÓTESIS Y OBJETIVOS ..... - 4 -

1.2.1 HIPOTESIS ..... *- 4 -*

1.2.2 OBJETIVOS ..... *- 5 -*

1.3 ANTECEDENTES ..... - 5 -

1.4 MARCO TEÓRICO ..... - 6 -

1.4.1 SERIES TEMPORALES ..... *- 6 -*

1.4.2 ALGORITMOS EVOLUTIVOS ..... *- 6 -*

1.4.3 ALGORITMOS GENÉTICOS ..... *- 7 -*

1.4.4 PROGRAMACIÓN GENÉTICA ..... *- 8 -*

1.5 METODOLOGÍA ..... - 10 -

1.5.1 DIMENSIÓN DE LA ESTRATEGIA GENERAL ..... *- 10 -*

1.5.2 DIMENSIÓN DE LAS TÉCNICAS DE OBTENCIÓN Y ANÁLISIS DE INFORMACIÓN EMPÍRICA ..... *- 11 -*

1.6 ESTRUCTURA DE LA TESIS ..... - 12 -

1.7 DISCUSIONES Y COMENTARIOS ..... - 13 -

2 CAPITULO II PROGRAMACIÓN GENÉTICA ..... **- 14 -**

2.1 INTRODUCCIÓN ..... - 15 -

2.2 ALGORITMOS GENÉTICOS ..... - 16 -

2.3 INDIVIDUOS ..... - 17 -

2.4 ALGORITMO GENÉTICO GENÉRICO ..... - 18 -

2.4.1 EXPLICACIÓN DEL ALGORITMO ..... *- 19 -*

2.5 PROGRAMACIÓN GENÉTICA ..... - 19 -

2.6 ESTADO DEL ARTE GP ..... - 20 -

2.7 REPRESENTACIÓN DE LOS PROGRAMAS ..... - 26 -

2.8 DIAGRAMA DE FLUJO DE LA PROGRAMACIÓN GENÉTICA ..... - 27 -

2.9 GENERACIÓN INICIAL ..... - 28 -
2.10 OPERADORES GENÉTICOS ..... - 29 -
2.10.1 CRUCE ..... *- 29 -*
2.10.2 MUTACIÓN ..... *- 30 -*
2.11 EVALUACIÓN ..... - 31 -
2.12 PARÁMETROS ..... - 31 -
2.13 DISCUSIONES Y COMENTARIOS ..... - 32 -

3 CAPITULO III PROGRAMACIÓN GENÉTICA LINEAL ..... - 33 -

3.1 INTRODUCCIÓN ..... - 34 -
3.2 ESTADO DEL ARTE LGP ..... - 35 -
3.3 CONCEPTOS BÁSICOS DE LA LGP ..... - 39 -
3.4 CARACTERÍSTICAS DE LA LGP ..... - 40 -
3.5 PASOS APLICADOS PARA LA ELABORACIÓN DEL LGP ..... - 40 -
3.6 REPRESENTACIÓN DE INDIVIDUOS ..... - 43 -
3.7 OPERADORES EVOLUTIVOS ..... - 45 -
3.7.1 CRUZAMIENTO LINEAL ..... *- 45 -*
3.7.2 MUTACIÓN LINEAL ..... *- 48 -*
3.8 ALGORITMO UTILIZADO ..... - 51 -
3.9 DISCUSIONES Y COMENTARIOS ..... - 54 -

4 CAPITULO IV METODOLOGÍA DE LAS PRUEBAS EXPERIMENTALES ..... - 55 -

4.1 INTRODUCCIÓN ..... - 56 -
4.2 METODO DE PROGRAMACIÓN GENÉTICA LINEAL ..... - 57 -
4.3 SERIES TEMPORALES ..... - 57 -
4.4 PATRONES DE SERIES DE TIEMPO ..... - 58 -
4.4.1 TENDENCIA ..... *- 59 -*
4.4.2 MOVIMIENTOS ESTACIONALES ..... *- 59 -*
4.4.3 MOVIMIENTO CICLICOS ..... *- 60 -*
4.4.4 MOVIMIENTOS IRREGULARES O ALEATORIOS ..... *- 61 -*
4.5 SERIES DE TIEMPO SELECCIONADOS PARA EL ESTUDIO ..... - 62 -
4.5.1 REGRESIÓN LINEAL ..... *- 62 -*
4.5.2 PROMEDIO MOVIL ..... *- 62 -*
4.5.3 SUAVIZADO EXPONENCIAL ..... *- 62 -*
4.5.4 SUAVIZADO EXPONENCIAL CON TENDENCIA ..... *- 63 -*
4.6 APLICACIÓN DE LA METODOLOGIA DESARROLLADA ..... - 64 -
4.6.1 DISTRIBUCIÓN DE DATOS ..... *- 64 -*
4.6.2 EQUIPO UTILIZADO ..... *- 64 -*
4.6.3 MÉTRICAS DE DESEMPEÑO ..... *- 65 -*
4.7 PROCEDIMIENTO DE EJECUCIÓN PARA CADA MÉTODO ..... - 65 -

4.8 ESTUDIO DE LOS RESULTADOS OBTENIDOS ..... - 67 -
4.8.1 RESULTADOS DEL LGP ..... *- 67 -*
4.8.2 MEJORES SOLUCIONES ..... *- 67 -*
4.8.3 RESULTADO CON EL INDIVIDUO A ..... *- 68 -*
4.8.4 RESULTADO CON EL INDIVIDUO B ..... *- 70 -*
4.8.5 RESULTADO CON EL INDIVIDUO C ..... *- 71 -*
4.8.6 RESULTADO DE LA REGRESION LINEAL ..... *- 73 -*
4.8.7 RESULTADO DEL PROMEDIO MÓVIL ..... *- 74 -*
4.8.8 SUAVIZADO EXPONENCIAL ..... *- 76 -*
4.8.9 SUAVIZADO EXPONENCIAL CON TENDENCIA ..... *- 77 -*
4.9 COMPARACIÓN Y EVALUACIÓN DE RESULTADOS ..... - 79 -
4.9.1 PRIMERA MÉTRICA DEL ERROR CUADRÁTICO MEDIO ..... *- 79 -*
4.9.2 SEGUNDA MÉTRICA DEL ERROR ABSOLUTO MEDIO ..... *- 80 -*
4.10 APRENDIZAJE DEL LGP ..... - 82 -
4.11 DISCUSIONES Y COMENTARIOS ..... - 83 -

5 CAPITULO V CONCLUSIONES Y LÍNEAS FUTURAS DE TRABAJO ..... - 84 -

5.1 RESULTADOS Y CONCLUSIONES ..... - 85 -
5.2 CONCLUSIONES METODOLÓGICAS ..... - 85 -
5.3 CONCLUSIONES DE TENDENCIA ..... - 86 -
5.4 CONCLUSIONES Y RESULTADOS DE RELEVANCIA ..... - 87 -
5.5 TRABAJOS FUTUROS ..... - 89 -

REFERENCIAS ..... - 90 -

ANEXO 1 ..... - 96 -

ANEXO 2 ..... - 98 -

ANEXO 3 ..... - 99 -

ÍNDICE DE FIGURAS

FIGURA 2-1 - INDIVIDUO GENÉTICO BINARIO [7] ..... - 17 -
FIGURA 2-2 - NODOS TERMINALES (5),(3),(X)Y(2) [8] ..... - 26 -
FIGURA 2-3 - NODOS FUNCIONALES (-),(*)Y(*) [8] ..... - 27 -
FIGURA 2-4–DIAGRAMA DE FLUJO PARA UN PROGRAMA GENÉTICO [8] ..... - 28 -
FIGURA 2-5 - INDIVIDUOS Y NODOS SELECCIONADOS PARA EL CRUCE [8] ..... - 29 -
FIGURA 2-6 - RESULTADO DEL CRUCE [8] ..... - 30 -
FIGURA 3-1 - CRUZAMIENTO EN PROGRAMACIÓN GENÉTICA LINEAL. FIGURA CREADA SEGÚN [1] ..... - 46 -
FIGURA 4-1 - SERIE DE TIEMPO DE INGRESOS DE CAUSAS PENALES [FUENTE PROPIA] ..... - 58 -
FIGURA 4-2 - DEMANDA CON TENDENCIA [FUENTE PROPIA] ..... - 59 -
FIGURA 4-3 - DEMANDA CON ESTACIONALIDAD [FUENTE PROPIA] ..... - 60 -
FIGURA 4-4 - DEMANDA CON ESTACIONALIDAD Y TENDENCIA [FUENTE PROPIA] ..... - 60 -
FIGURA 4-5 - DEMANDA ESTACIONARIA [FUENTE PROPIA] ..... - 61 -
FIGURA 4-6 - DEMANDA ESTACIONARIA [FUENTE PROPIA] ..... - 61 -
FIGURA 4-7 - RESULTADO DE LA EJECUCIÓN 38 DEL LGP [FUENTE PROPIA] ..... - 69 -
FIGURA 4-8 - RESULTADO GRÁFICO EJECUCIÓN 39 LGP [FUENTE PROPIA] ..... - 71 -
FIGURA 4-9 - PREDICCIÓN EJECUCIÓN 24 DEL LGP [FUENTE PROPIA] ..... - 72 -
FIGURA 4-10 - RESULTADO REGRESIÓN LINEAL [FUENTE PROPIA] ..... - 74 -
FIGURA 4-11 - RESULTADO PROMEDIO MÓVIL [FUENTE PROPIA] ..... - 75 -
FIGURA 4-12 - RESULTADO SUAVIZADO EXPONENCIAL [FUENTE PROPIA] ..... - 77 -
FIGURA 4-13 - RESULTADO SUAVIZADO EXPONENCIAL CON TENDENCIA [FUENTE PROPIA] ..... - 79 -
FIGURA 4-14 - DEMOSTRACIÓN DEL APRENDIZAJE DEL LGP [FUENTE PROPIA] ..... - 82 -
FIGURA 5-1 - TENDENCIA DE CRECIMIENTO DE LOS INGRESOS [FUENTE PROPIA] ..... - 86 -

ÍNDICE DE TABLAS

TABLA 3-1 - REGISTRO DEFINIDOS PARA LA LGP [FUENTE PROPIA] ..... - 41 -
TABLA 3-2 - OPERACIONES PARA LA LGP [1] ..... - 42 -
TABLA 3-3 - PARÁMETROS UTILIZADOS EN LA LGP [FUENTE PROPIA] ..... - 43 -
TABLA 3-4 - SEGMENTOS SELECCIONADOS PARA EL CRUCE. TABLA CREADA SEGÚN [1] ..... - 47 -
TABLA 3-5 - RESULTADO DEL CRUCE. TABLA CREADA SEGÚN [1] ..... - 48 -
TABLA 4-1 - MEJORES INDIVIDUOS DE LAS EJECUCIONES DEL LGP [FUENTE PROPIA] ..... - 68 -
TABLA 4-2 - RESULTADOS DE LA EJECUCIÓN 38 DEL LGP [FUENTE PROPIA] ..... - 69 -
TABLA 4-3 - RESULTADOS DE LAS MÉTRICAS LGP EJECUCIÓN 39 [FUENTE PROPIA] ..... - 70 -
TABLA 4-4 - RESULTADOS DE LAS MÉTRICAS LGP EJECUCIÓN 39 [FUENTE PROPIA] ..... - 71 -
TABLA 4-5 - RESULTADO DE LAS MÉTRICAS REGRESIÓN LINEAL [FUENTE PROPIA] ..... - 73 -
TABLA 4-6 - RESULTADO DE LAS MÉTRICAS PROMEDIO MÓVIL [FUENTE PROPIA] ..... - 75 -
TABLA 4-7 - RESULTADO DE LAS MÉTRICAS SUAVIZADO EXPONENCIAL [FUENTE PROPIA] ..... - 76 -
TABLA 4-8 - RESULTADOS DE LAS MÉTRICAS SUAVIZADO EXPONENCIAL CON TENDENCIA [FUENTE PROPIA] .... - 78 -
TABLA 4-9 - COMPARACIÓN DE RESULTADOS CON ERROR CUADRÁTICO MEDIO [FUENTE PROPIA] ..... - 80 -
TABLA 4-10 - COMPARACIÓN DE RESULTADOS CON ERROR ABSOLUTO MEDIO [FUENTE PROPIA] ..... - 81 -
TABLA 5-1 - TABLA DE POSICIONAMIENTO DE LA PRIMERA MÉTRICA [FUENTE PROPIA] ..... - 87 -
TABLA 5-2 - TABLA DE POSICIONAMIENTO DE LA SEGUNDA MÉTRICA [FUENTE PROPIA] ..... - 88 -

LISTA DE ACRÓNIMOS

EA	*Evolutionary Algorithm*.
EP	*Evolutionary Programming*.
GA	*Genetic Algorithm*.
ES	*Evolutionary Strategies*
GP	*Genetic Programming*.
LGP	*Linear Genetic Programming*.

LISTA DE SÍMBOLOS

m	Número de individuos por población para un EA.
n	Número de generaciones para un EA.
I	Mejor individuo de la población.
T	Población en un EA.
xi	Individuo a partir del cual se forma el nuevo individuo en una EA.
xj	Valor real para el periodo j.
pj	Pronóstico para el periodo j.
N	Número de muestras u observaciones.
Tmax	Tamaño máximo de un programa en LGP.
Tmin	Tamaño mínimo de un programa en LGP.
Y(t)	Variables aleatorias definidas sobre un espacio de probabilidad.
Yt	Valores de una serie temporal.
Tt	Tendencia en el periodo t.
St+1	Estimación para el periodo t+1.
α	Constante de suavizado.
β	Constante de suavizado para la tendencia.
S0	Período inicial de estimación.
max_gen	Máximo número de generaciones para un programa LGP.
max_pob	Máximo número de individuos para una generación en un programa LGP.
mejorInd	Mejor individuo hallado por un programa en LGP.
r[i]	Registros.
e^2	Error al cuadrado.
|e |	Error absoluto.
P(A)	Pronóstico con el Individuo A.

P(B) Pronóstico con el Individuo B.

P(C) Pronóstico con el Individuo C.

P(Regresión) Pronóstico de la Regresión Lineal.

P(P. Movil) Pronóstico del Promedio Móvil.

P(S.Expon) Pronóstico del Suavizado Exponencial.

P(S.Exp.Tend.) Pronóstico del Suavizado Exponencial con Tendencia.

RESUMEN

Este trabajo propone una metodología de predicción de ingresos de causas penales utilizando una variación de la Programación Genética GP, la Programación Genética Lineal LGP. El estudio se realizó en base a datos mensuales recogidos durante seis años (2007 a 2013), de los siete Juzgados Penales de Garantías de Ciudad del Este.

La verificación del método propuesto se hizo mediante la comparación con modelos estadísticos, por lo que se ha estimado la misma serie de tiempo con estos modelos. La validación de los modelos fueron realizados mediante dos métricas; el error cuadrático medio y el error absoluto medio.

El método LGP generó varios resultados gracias a su capacidad aleatoria de crear fórmulas matemáticas, de las mismas tres resultados importantes fueron seleccionados para su comprobación. Dos resultados quedaron en primer y segundo lugar, ganándole así a los métodos estadísticos, y el tercer resultado seleccionado no obtuvo una buena predicción, pero sí un buen trazado en el gráfico comparándolo con la serie de tiempo. La conclusión en base a los resultados obtenidos, indican que los modelos generados por LGP son capaces de pronosticar con una buena precisión los ingresos penales, logrando mejores resultados que los métodos estadísticos.

Palabras clave: Programación genética, programación genética lineal, ingresos penales

ABSTRACT

This paper proposes a methodology for predicting admissions of criminal cases using a variation of Genetic Programming GP, Linear Genetic Programming LGP. It has been studied based on monthly data collected during six years (2007-2013) of the seven Criminal Courts of Ciudad del Este.

Verification of the proposed method was carried out by comparison with statistical models, so it has been estimated time series with these models. The validation of the models was made using two metrics; the mean square error and the mean absolute error.

The LGP method repeatedly results from its ability to create random math formulas, the same three important results were selected for testing. Two results were in first and second place, beating and statistical methods, and the third selected result did not obtain a good prediction, but a good path in the graph compared to the time series. The conclusion based on the results obtained indicate that LGP generated models are able to predict criminal income with good accuracy achieved better results than statistical methods.

Keywords: Genetic programming, linear genetic programming, criminal income.

CAPITULO I
INTRODUCCIÓN

1.1 INTRODUCCIÓN

La utilización de inteligencia artificial en el área de las ciencias jurídicas para realizar investigaciones es poco común. La naturaleza jurídica es el estudio de normas y leyes aplicables al derecho, basadas en libros escritos por antiguas civilizaciones. En consecuencia, el profesional de esta área está familiarizado con libros y códigos impresos y no necesariamente con los últimos avances tecnológicos en materia de inteligencia artificial. Esta situación con los profesionales del derecho conlleva una oportunidad de realizar aportes relevantes, considerando la inmensa cantidad de información disponible.

En este trabajo se utilizaron datos netamente numéricos de una situación concreta de la problemática actual para estudiar el crecimiento de la delincuencia en una región bien limitada del país, Ciudad del Este.

El crecimiento denotado aparenta ser gradual y en la actualidad los Juzgados Penales se encuentran abarrotados de expedientes que se generan a partir de los delitos cometidos en Ciudad del Este. Este fenómeno generó una necesidad de análisis, a los efectos de detectar cuál sería la magnitud de este crecimiento, de forma a planificar acciones futuras.

En la investigación se ha aplicado una predicción utilizando la programación genética lineal así como métodos estadísticos. El objetivo es lograr estimaciones razonables y para ello se elaboró una metodología en base a los resultados, de forma que puedan ser aplicados en cualquier región del país.

La predicción razonable del número de delitos, es un aporte relevante para alertar a las autoridades a modo de prever medidas eficientes para manejar la situación.

1.2 HIPÓTESIS Y OBJETIVOS

1.2.1 HIPOTESIS

La aplicación de la Programación Genética Lineal como metodología de predicción de ingresos de causas penales, mejora significativamente la estimación de situaciones futuras sobre la cantidad de delitos que se pudieran cometer.

1.2.2 OBJETIVOS

1.2.2.1 PARA LA ACCIÓN

Objetivo *general*:

A. Desarrollar mediante la aplicación de la programación genética lineal una metodología para la predicción de ingresos de causas penales y compararlos con métodos estadísticos.

1.2.2.2 DE CONOCIMIENTO

Objetivos *específicos*:

a. Construir un programa genético lineal, que realice la predicción de los ingresos de causas penales.
b. Comparar los resultados de la programación genética lineal, con métodos estadísticos.
c. Desarrollar una metodología que pueda ser aplicada en cualquier región del país para predecir los ingresos de causas penales.
d. Determinar la tendencia de los ingresos, para años posteriores a los estudiados.
e. Determinar si la infraestructura actual, será suficiente para años posteriores a los estudiados.
f. Determinar el dimensionamiento adecuado en el área de recurso humano.
g. Determinar informaciones relevantes que se puedan lograr con este estudio.

1.3 ANTECEDENTES

En el área de investigación judicial, no se ha encontrado publicaciones científicas aplicando programación genética lineal. En vista a esta situación, se ha recurrido a observar trabajos similares aplicados a otras áreas como ser las económicas y financieras. Por ejemplo, publicaciones que obtengan reglas de mercado que permitan estimar algunos indicadores económicos como el Índice de Precios al Consumidor y el Producto Interno Bruto además de otros indicadores y series de tiempos aplicando LGP [1].

La programación genética ya es bastante utilizada en aplicaciones financieras, y de hecho las metodologías de pronósticos para el comercio son muy importantes. Realizar pronósticos adecuados y confiables posibilita la conveniencia de efectuar actividades económicas. Así, se aplicó la programación genética al mercado de intercambio de divisas [2].

El pronóstico de los mercados financieros se emplea para predecir la volatilidad, en este caso se utilizó la programación genética aplicando limitaciones en la sintáctica de los árboles [3].

1.4 MARCO TEÓRICO

1.4.1 SERIES TEMPORALES

Los métodos de serie temporal utilizan datos históricos como base para estimar resultados futuros [4]. Se asume que la demanda es en función del tiempo, y que además pueden estar involucrados los siguientes componentes:

- Tendencia.
- Ciclos.
- Estacionalidades.
- Irregularidades.

Una serie de tiempo es una serie de observaciones temporales de alguna cantidad de interés [5]. El análisis de una serie de tiempo se realiza con el objetivo de conocer su patrón de comportamiento de forma a prever su evolución en un futuro cercano.

Las predicciones, son métodos con los cuales, se puede ayudar a la mejora de acciones de control o de toma de decisiones en entornos industriales, sociales y económicos. Es necesario tener métodos que puedan adaptarse a condiciones cambiantes, posibilitando pronósticos de series de tiempo más adecuados, motivo por el cual se ha optado en esta investigación por utilizar la programación genética lineal por ser una metodología comprobada por series de investigaciones como una de las mejores para realizar predicciones futuras[10].

En el área específica de ciencias jurídicas, no se ha encontrado ninguna investigación preliminar, en base a los métodos de predicción que se quiere realizar en este proyecto. Existen trabajos similares aplicados en otras áreas que ayudarán a encaminar este proyecto, y que arrojan resultados alentadores [11].

1.4.2 ALGORITMOS EVOLUTIVOS

Los algoritmos evolutivos (EA – *Evolutionary Algorithm*) son métodos de búsqueda colectiva en un espacio de soluciones. Dada una población inicial de potenciales soluciones a un problema, el algoritmo evoluciona esta población a nuevas y posibles mejores soluciones.

Cada individuo es representado como un cromosoma, y con ellos se realizan las operaciones evolutivas para producir las siguientes generaciones.

Los EA siguen la teoría Darwiniana, "la supervivencia del más fuerte". Existe una población de individuos que se reproducen de diversas formas, unos con otros. Los mejores individuos sobrevivirán y ocasionalmente evolucionarán [6].

Esta evolución de individuos se realiza aplicando de manera iterativa las operaciones genéticas o evolutivas sobre los individuos de la población. Antes de realizar cualquier operación genética se suele realizar un proceso de selección que consiste en la identificación de aquellos individuos a los cuales se les aplicará las operaciones de cruzamiento, mutación y reproducción.

A continuación se detallan estas operaciones:

• Cruzamiento: para aplicar esta operación se necesitan 2 individuos que serán los padres que aportarán el material genético para los hijos. Aleatoriamente se escoge un punto de cruzamiento donde se intercambian las estructuras de los padres para formar los nuevos individuos.

• Mutación: Se modifica aleatoriamente la estructura del individuo padre para generar el nuevo individuo.

• Reproducción: consiste en la identificación de los mejores individuos de la población actual que conformarán la nueva población de manera a no perder las mejores soluciones encontradas hasta el momento. Este proceso también es conocido como Elitismo.

1.4.3 ALGORITMOS GENÉTICOS

Algoritmo es una serie de representación de pasos organizados que pueden describir un proceso a seguir, a los efectos de dar solución a un problema específico. Una de las técnicas evolutivas de mayor utilización en la actualidad son los Algoritmos Genéticos (GA – *Genetic Algorithm*) que fueron propuestos por *Holland* [7]. Este proceso evolutivo se basa en la aplicación de mecanismos computacionales inspirados en la evolución de las especies, tratando de mejorar la información genética de los individuos aplicando operaciones como selección, cruzamiento y mutación sobre un conjunto de potenciales soluciones a un problema dado.

Características de los algoritmos genéticos:

- La codificación más común de los individuos es a través de cadenas binarias, aunque se han utilizado también números reales y letras. El primero se utiliza generalmente debido a que es el método que propuso originalmente *Holland* [7], y además resulta muy sencillo de implementar.
- La aplicación más común de los algoritmos genéticos ha sido la solución de problemas de optimización.
- La función de aptitud no es más que la función objetivo del problema de optimización a ser estudiada. El algoritmo genético únicamente maximiza, pero la minimización puede realizarse fácilmente utilizando el recíproco de la función. Una característica que debe tener esta función es que tiene ser capaz de identificar a las malas soluciones, y de evolucionar a las buenas, de forma que estas últimas sean las que generen los mejores resultados.
- Los Algoritmos Genéticos resuelven los problemas generando poblaciones sucesivas a las que se aplican los operadores de mutación y cruce. Cada individuo representa una solución al problema, y se trata de encontrar al individuo que represente a la mejor solución.

Es importante saber que antes de poder aplicar los algoritmos genéticos se deben definir algunos componentes necesarios para el proceso [8]:

- Representación de soluciones.
- Población inicial.
- Función de *fitness*.
- Operaciones genéticas.
- Parámetros generales.
- Condición de fin.

1.4.4 PROGRAMACIÓN GENÉTICA

La naturaleza utiliza una forma de evolucionar organismos donde los menos aptos para el ambiente tienden a perecer mientras los más capacitados viven y se reproducen para generar nuevos individuos. Los hijos tienen material genético de los padres y ocasionalmente ocurren mutaciones en estos individuos, y aunque algunos de ellos mueren, otros individuos evolucionan mejorando la especie [7].

La computación evolutiva se inspira en el principio natural de la evolución de las especies. Algunas de sus características son:

• Los individuos son programas informáticos, tradicionalmente representados en la memoria como estructuras de árboles [9].Los árboles pueden ser fácilmente evaluados de forma recursiva. Cada nodo del árbol tiene una función como operador y cada nodo terminal tiene un operando, por lo que las expresiones matemáticas son fáciles de evolucionar y evaluar.

• Existen individuos que poseen la posibilidad de reproducirse con una cierta probabilidad. Cada individuo representa una solución al problema que se desea resolver. Cada vez que se reproducen estas soluciones, se realiza una exploración del espacio de soluciones buscando aproximarnos al óptimo.

• Cada individuo tiene asociado un valor de adaptabilidad que nos indica que tan buena es la solución para el problema en cuestión. Este valor es utilizado a la hora de seleccionar los individuos que producirán la nueva generación.

• Todos los individuos son agrupados en un conjunto de soluciones al cual se denomina población. Mediante las operaciones evolutivas esta población se renueva en cada generación buscando soluciones cada vez mejores.

• Dentro de la población existe una diversidad entre sus individuos de forma a tener un muestreo representativo del espacio de búsqueda. Esto nos asegura una mejor exploración del espacio de soluciones.

La Programación Genética, consiste en la evolución automática de programas usando ideas fundamentadas de la selección natural, permitiendo realizar regresión simbólica, o sea puede obtener además de un dato numérico predictivo, una expresión matemática en función de las variables de entrada, que trata de identificar al sistema estudiado en lo que hace respecto del proceso modelado. La Programación Genética es una técnica de Minería de Datos y es considerada como la más prometedora en el área de inducción de expresiones interpretables, demostrando precisiones muy competitivas [10].

1.5 METODOLOGÍA

1.5.1 DIMENSIÓN DE LA ESTRATEGIA GENERAL

1.5.1.1 TIPO DE ESTRATEGIA Y FUNDAMENTACIÓN

Se realizó un estudio de series temporales aplicando programación genética lineal y métodos estadísticos, para lograr la metodología de predicción más adecuada de los ingresos de causas penales de los siete Juzgados de Garantías de Ciudad del Este, en el periodo de tiempo enero/2007 a diciembre/2013.

1.5.1.2 UNIVERSO

El *universo* está constituido por los registros de causas penales de Ciudad del Este correspondientes a los Juzgados de Garantías, durante el periodo de enero/2007 a diciembre/2013 (aproximadamente 38.305 registros).

1.5.1.3 UNIDAD DE ANÁLISIS

La *unidad de análisis* está integrada por *cada uno de los Juzgados de Garantías Penales* de Ciudad del Este de la Sexta Circunscripción Judicial, durante los años 2007 al 2013.

1.5.1.4 SELECCIÓN DE CASOS

Los casos seleccionados son *la cantidad total de ingresos divididos en periodos de un mes*, desde Enero/2007 hasta Diciembre/2013, totalizando 84 meses. (Ver *Anexo 1*)

1.5.1.5 ÉNFASIS EN EL PROCESO LINEAL

Este trabajo de investigación es del tipo cuantitativo, y fue realizado mediante el análisis de datos numéricos. La cantidad de ingresos de causas penales son los datos principales de la investigación, los cuales fueron estudiados, analizados y comparados, mediante metodologías estadísticas y de inteligencia artificial.

1.5.1.6 ROL DEL INVESTIGADOR

En una *lógica cuantitativa* como la adoptada para la presente investigación, el investigador intenta descubrir un resultado adecuado, aplicando en un sector ciertas metodologías, para posteriormente poder ser aplicado en un universo mayor.

1.5.2 DIMENSIÓN DE LAS TÉCNICAS DE OBTENCIÓN Y ANÁLISIS DE INFORMACIÓN EMPÍRICA

1.5.2.1 TÉCNICAS DE OBTENCIÓN Y ANÁLISIS

Fase 1

1) Generar y obtener los datos: Los datos fueron obtenidos a partir de los sitios web de la Excma. Corte Suprema de Justicia [44] y del Poder Judicial [45], que se encuentran en línea y son de libre acceso al público en general.
2) Realizar investigaciones bibliográficas: Búsqueda de libros, publicaciones, investigaciones de referencia. Selección de materiales para el marco teórico.
3) Definir qué porción de datos serán utilizadas para entrenamiento y que porción para la validación.

Fase 2

4) Desarrollar el programa genético: En base a las investigaciones bibliográficas y los algoritmos correspondientes a ser aplicados, se creó el programa genético lineal, encargado de generar las predicciones a ser estudiadas en el presente proyecto.
   La metodología seleccionada es la evolutiva computacional, llamada programación genética lineal que es una línea de investigación de los algoritmos evolutivos. El Algoritmo Evolutivo, es un modelo computacional de la evolución biológica, que se basa en los principios de la evolución y la herencia.
   Características del método:
   - Los Algoritmos Evolutivos seleccionan una buena solución sobre un conjunto de soluciones denominado población, de las cuales se puede obtener el mejor individuo.
   - Se utiliza una función de adaptabilidad (*fitness*) para comparar soluciones.
5) Utilizar el programa y varios métodos estadísticos con los datos obtenidos: introducir los datos en el programa para su evaluación. Al construir un modelo utilizando los Algoritmos Evolutivos, se ha utilizado varias herramientas como: diagramas de flujo y pseudocódigos (algoritmos y operadores genéticos). Al tener desarrollado el programa, se ejecutó el algoritmo para obtener los resultados experimentales. Para comparar los resultados, se aplicaron los mismos datos a métodos estadísticos, estos son: regresión lineal, promedio móvil, suavizado exponencial y suavizado exponencial con tendencia.

Fase 3

6) Estudiar los resultados obtenidos: Se verificaron los resultados generados de los estudios realizados y las conclusiones preliminares.
7) Evaluar los resultados experimentales finales.

Fase 4

8) Conclusión: Formalizar los resultados en el libro de tesis y artículos científicos.

1.6 ESTRUCTURA DE LA TESIS

CAPÍTULO I - INTRODUCCIÓN

Introducción a la problemática planteada con una descripción de la situación. Encuadre del marco teórico que sirve de introducción al contenido que se estudia. Hipótesis y objetivos del estudio, así también la estrategia de investigación implementada y las técnicas aplicadas para la obtención y análisis de la información.

CAPÍTULO II–PROGRAMACIÓN GENÉTICA

Conceptos generales sobre los Algoritmos Genéticos y la Programación Genética, dando un enfoque a lo que es el estudio del área.

CAPÍTULO III–PROGRAMACIÓN GENÉTICA LINEAL

Formulación y aplicación práctica de la LGP, características principales, pasos aplicados para su elaboración, representación de los individuos.

CAPÍTULO IV–METODOLOGÍA DE LAS PRUEBAS EXPERIMENTALES

Metodología de la aplicación de los métodos creados y de los métodos estadísticos con sus respectivas fórmulas. Estudio y comparación de los resultados.

CAPÍTULO V–CONCLUSIONES GENERALES

Resultados y conclusiones, Conclusiones Metodológicas, Conclusiones de Tendencia, Conclusiones Finales y Resultados de Relevancia.

1.7 DISCUSIONES Y COMENTARIOS

Los algoritmos genéticos fueron profundizados para escribir este manual de tesis, pasando a entender su funcionamiento en todo su ámbito práctico. Asimismo se investigó sobre la programación genética que es también un área de estudio de los algoritmos evolutivos, a los efectos de conocer sus procesos y fases para su aplicación.

Adentrándose a la Programación Genética Lineal, se vio que es una aplicación compleja, pero con resultados óptimos para el área que se utiliza [10].Entonces se puede decir que la metodología seleccionada permite desarrollar sistemas inteligentes, que aplican técnicas de aprendizaje de máquina, las cuales se utilizaron para crear una aplicación que trabaja sobre una de serie de tiempo con datos históricos perteneciente a un área jurídica. Esta técnica es un camino efectivo y eficiente para crear modelos que resuelvan los fenómenos similares a los estudiados.

El modelo seleccionado consiste en la evolución automática de programas que usa ideas basadas en la selección natural, y permiten obtener como resultado además de algún dato numérico predictivo, una expresión matemática en función a las variables de entrada.

Esta técnica está comprobada por varios investigadores hoy en día, y resulta ser muy promisoria en la creación de expresiones interpretables y a la vez demuestra bastante precisión competitiva en el área que se la aplica [10].

El área de estudio de los Algoritmos Evolutivos se encuentra dividida en las siguientes corrientes de investigación [8] y [55]:

- Algoritmo genético.
- Estrategias evolutivas.
- Programación evolutiva.

CAPITULO II
PROGRAMACIÓN GENÉTICA

2.1 INTRODUCCIÓN

En esta sección se comienza explicando los algoritmos genéticos y la programación genética, la definición de estas técnicas orientan el principio de la aplicación de la metodología del proyecto. La teoría de la evolución de Darwin [6] es empleada en programas de computadores, haciéndoles que sean capaces de evolucionar como lo hace un organismo vivo [11].

Las descripciones sobre las bases, fundamentos y funcionamiento de estas técnicas se incluyen en el material, desde la creación de los individuos, su evolución a través de operadores genéticos como el cruzamiento, la mutación y la reproducción, y la solución a través de esta metodología de problemas de optimización.

La aplicación práctica de estos algoritmos, se incluyó mediante un algoritmo genético genérico, conteniendo la explicación de cada paso que realiza este algoritmo durante su ejecución.

La GP se estudió ilustrando cada fase para conocer el proceso y su aplicación técnica. Como todo algoritmo evolutivo, se emplea la evolución en este caso de programas automatizados que se desarrollan en cada población que se genera, simulando lo que ocurre en los organismos biológicos.

Los avances sobre este tema son gracias a las múltiples investigaciones que se realizan en el mundo entero. La computación evolutiva se utiliza en la actualidad en muchos y diferentes problemas cotidianos, y las áreas donde más se aplican son: estudios de optimización, tabla de enrutamiento, predicciones, diseños automatizados, calibraciones, hallazgo de errores, selección óptima de modelos matemáticos, planificación de producción, etc. [10].

2.2 ALGORITMOS GENÉTICOS

Los algoritmos genéticos son métodos adaptativos, que generalmente se usan en problemas de búsqueda y optimización [11], estos métodos se basan en la teoría de la evolución de las especies de Darwin [6], donde se observa que la naturaleza tiene una forma de evolucionar a los organismos, donde los menos aptos mueren, mientras los más aptos siguen la vida y se reproducen para generar nuevos individuos.

Los GA son algoritmos de búsqueda que se basan en la selección natural y la genética natural [7], donde los hijos heredan material genético de los padres y en ciertos casos pueden ocurrir mutaciones, pero aunque alguno de estos individuos puede morir, otros sobreviven evolucionando y en consecuencia, mejorando la especie.

El método evolutivo se utiliza para hallar la solución a un problema, el procedimiento se inicia con un conjunto primario de individuos, al cual se lo denomina población inicial, que habitualmente se genera de forma aleatoria. Cada individuo representa una posible solución al problema. Estos individuos deben evolucionar, generando nuevas poblaciones; cada generación debe de adaptarse mejor al resultado, y tras el paso de cada una de las nuevas generaciones, se espera llegar a una buena solución.

Los GA trabajan con una población de individuos, donde cada individuo representa una posible solución al problema. Estas soluciones constan de una estructura que se lo denomina cromosoma. Cada individuo de la población tiene un valor de ajuste o *fitness*, encargado de cuantificar su validez como solución al problema. Este valor es importante, ya que de ello dependerán sus oportunidades de reproducción.

Dentro del proceso natural no se conoce en su totalidad hoy en día, pero si se maneja las bases con ciertas incógnitas a descubrir. En base a los experimentos se puede saber que [6] y [7]:

- La evolución opera más sobre los cromosomas, que sobre los propios individuos. Los cromosomas se pueden considerar como herramientas que codifican la vida.

- La selección natural relaciona a los cromosomas (genotipo) con la eficiencia respecto a su entorno (fenotipo) que representa. Esto da la oportunidad a los individuos mejores adaptados de tener una mayor capacidad de reproducirse.

- La evolución tiene su lugar durante la reproducción. A este proceso le afecta una serie de mecanismos como son el cruce, la reproducción y la mutación.

2.3 INDIVIDUOS

La solución potencial a un problema se presenta dando unos valores a una serie de parámetros. Este conjunto de parámetros es denominado "genes", a cuya codificación en cadena de valores se le asigna el nombre de cromosoma.

El conjunto de parámetros representados por un cromosoma se lo conoce como genotipo. Este genotipo contiene información necesaria para llegar a la solución real del problema denominado fenotipo. En términos biológicos, la información genética contenida en el ADN de un individuo es el genotipo, mientras que la expresión de ese ADN (o sea el propio individuo) sería el fenotipo.

La codificación suele hacerse generalmente mediante valores binarios [7]. Se asigna un número de bits a cada parámetro y se realiza la discretización de la variable representada por cada gen. El número de bits asignado dependerá del grado de ajuste que se desea alcanzar. No todos los individuos tienen que tener el mismo número de bits. Cada bit perteneciente al gen suele recibir el nombre de alelo (*Ver figura 2-1*).

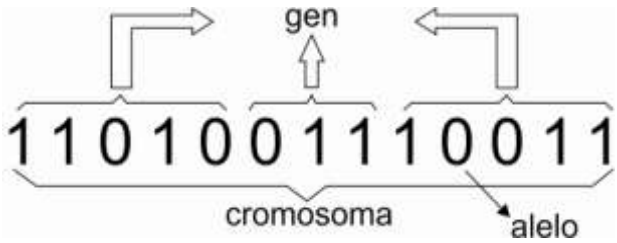


Figura 2-1 - Individuo genético binario [7]

Los Algoritmos Genéticos operan sobre una población de individuos y cada uno de estos individuos representa una posible solución al problema que se plantea. Todo individuo tiene asignado un valor de *fitness* con relación a la solución que representa.

La teoría Darwiniana de "la supervivencia del más fuerte" [5], trata de evolucionar los individuos con mayor *fitness*, y la evolución de los mismos se logra aplicando operaciones

genéticas o evolutivas. Una generación se obtiene a partir de una población anterior por medio de los operadores de reproducción y estos son:

- Cruzamiento: se crea la descendencia a partir del mismo número de individuos (generalmente 2) de la anterior generación.

- Reproducción: generalmente se identifican a los mejores individuos de la población y estos pasan a formar parte de la siguiente generación, sin sufrir cambios, evitando así la pérdida de las mejores soluciones.

- Mutación: se obtiene al modificar aleatoriamente la información genética de los individuos para formar a nuevos individuos, potencialmente mejores.

2.4 ALGORITMO GENÉTICO GENÉRICO

Un algoritmo es un conjunto de pasos que se encuentran organizados a los efectos de describir un proceso a seguir para dar solución a un problema específico.

Algoritmo 2.1: *Algoritmo Genético Genérico*

1: Inicializar aleatoriamente una población de individuos (soluciones).
2: Aleatoriamente seleccionar individuos de la población y compararlos de acuerdo a su *fitness*. La medida del *fitness* define la calidad de una solución en el problema que el algoritmo espera resolver.
3: Modificar individuos seleccionados utilizando alguno de los siguientes operadores de variación:
   - Reproducción. Copia un individuo sin cambiarlo.
   - Cruzamiento. Cambia subestructuras entre dos individuos.
   - Mutación. Cambia una sola unidad en un individuo en una posición aleatoria.
4: Se construye una nueva población y se calcula el valor de *fitness* de los nuevos individuos.
5: Si el criterio de fin no se cumple, volver a 2
6: Parar. El mejor individuo representa la mejor solución encontrada.

Este algoritmo genérico muestra el proceso de la evolución de los individuos de una población, que se genera en forma aleatoria, aplicándoles acciones similares a la evolución biológica, como son la mutación, el cruzamiento y la reproducción.

El uso práctico se empieza con la inicialización de las variables de entrada, que definen el número de individuos que tendrá cada población y la cantidad de generaciones, esta última podrá ser utilizada como criterio de parada del algoritmo. Con la entrada y el criterio de parada definidos, también es muy importante establecer la salida, que sería en este caso el mejor individuo hallado por el programa durante su ejecución.

2.4.1 EXPLICACIÓN DEL ALGORITMO

En la línea 1 del algoritmo se crea una población inicial de individuos generados al azar. Esta primera población es evaluada, seleccionando cada individuo de la población (línea 2) y utilizando su valor de *fitness* para determinar qué tan buena es la solución representada por el individuo para el problema que se quiere resolver.

En la línea 3, a los individuos seleccionados se les aplicarán los operadores genéticos y que darán a luz a la nueva población. Los métodos de selección más utilizados son el torneo y la ruleta [8].

En cada generación se crea una nueva población y se evalúa a cada individuo (línea 4) y mientras no se llega al criterio de parada (línea 5), se vuelve a la línea 2.

Si el criterio de parada se cumple, el mejor individuo representa la mejor solución al problema en cuestión (línea 6).

2.5 PROGRAMACIÓN GENÉTICA

La computación evolutiva es un conjunto de modelos computacionales, que desarrollan automáticamente programas de computadoras que evolucionan generación tras generación y que son considerados como probables soluciones de un problema determinado.

La GP es una técnica de aprendizaje automático que se utiliza para optimizar una población de programas de acuerdo a la función de ajuste o aptitud (*fitness*), que sería el responsable de evaluar la capacidad de cada programa.

En la naturaleza, los organismos tienen su forma de evolucionar, los más aptos viven y se reproducen para formar nuevos individuos. De esta reproducción nacen los hijos que tienen material genético de los padres. En ciertos casos estos individuos sufren mutaciones y aunque exista riesgo de deceso, la mayoría de los individuos evolucionan con el fin de mejorar la especie. La programación genética realiza este mismo procedimiento natural aplicándolo en

programas de computadores, y con este método se pretende resolver problemas mediante la conjunción de programas y algoritmos que los resuelven.

La potencialidad de la programación genética reside en que permite desarrollar programas de forma automatizada. Su base biológica es la misma que los algoritmos genéticos, la diferencia radica en la forma que se codifican los problemas.

2.6 ESTADO DEL ARTEGP

En [14], se ha comparado dos métodos de *soft-computing*las redes neuronales artificiales (RNA) y la programación genética (GP), para la producción de métodos que son capaces de predecir si una empresa va a tener resultados contables negativos. Con el fin de construir modelos de predicción que se pueden aplicar a un gran número de casos prácticos, se necesita crear modelos simples que requieren una pequeña cantidad de datos. En su trabajo han fusionado la capacidad de generación de árboles de clasificación del GP, con la capacidad de extracción de estructuras de la red neuronal no supervisada SOM (*Kohone`s self-organizing map*), utilizada como herramienta de *clustering,* logrando una herramienta de clasificación que superó los resultados obtenidos con un método RNA evolutivo.

El estudio del sistema adaptativo (AS) tuvo como objetivo proporcionar servicios adaptados a los usuarios en diferentes contextos. La investigación presento un método adaptativo basado en el principio de la programación genética (GP) mediante la extracción de las mejores normas de adaptación. Su idea fue de extraer reglas de adaptación interesantes y aplicarlas para adaptar la interfaz a cada usuario. Se utilizó GP por ser un método evolutivo para controlar la evolución del sistema y asegurar la adaptación de la población hacia las condiciones cambiantes (Adaptación, reactividad y tomando en cuenta el medio ambiente) [15].

En el artículo [16], se propone una nueva metodología para la predicción de series de tiempo no lineales usando programación genética. La aproximación propuesta se basa en la incorporación del concepto de bloques funcionales y la modificación del algoritmo genético para que opere con éstos. Según manifiesta, los bloques funcionales representan modelos estadísticos bien conocidos para el pronóstico de series de tiempo. El algoritmo propuesto permite la exploración y explotación de regiones donde hay mayor posibilidad de encontrar mejores modelos de pronóstico. Para validar la aproximación propuesta, se pronosticaron dos series de tiempo *Benchmark*; se encontró que la metodología pronostica con mayor precisión las series de tiempo consideradas en comparación con otros modelos no lineales.

Raffo [17] muestra la implementación de algoritmos de evolución diferencial en el software MATLAB®. Explicando que el precio que se paga por una optimización numérica eficiente, significa hacer uso de matemáticas complejas, sin embargo la evolución diferencial, es una excepción porque es fácil de aplicar y robusto en la optimización numérica. La evolución diferencial es una herramienta de diseño de gran utilidad en las aplicaciones prácticas. Afirma que el programa que presentó, es una implementación mejorada y eficiente al programa que presentan *Price* y *Storm*.

Otro material visto, fue la aplicación sobre el diseño, implementación y evaluación de un algoritmo que, a través de técnicas de programación genética y estrategias evolutivas, sea capaz de obtener una secuencia de números, dado una función al que describa en forma ajustada, sin partir de una información previa fuera de los elementos de la serie. Su objetivo general fue de analizar y aproximar funciones complejas y secuencia matemáticas notables, al estilo de la regresión clásica, aprovechando la flexibilidad de los algoritmos para evaluar simultáneamente funciones con formas dispares [18].

Varias áreas son seleccionadas para la implementación de la GP, en esta investigación, se ha estudiado los índices de vegetación (IVs), siendo este el método más utilizado para extraer información de la vegetación mediante imágenes satelitales. El investigador afirma que los modelos de erosión como la "Ecuación Universal Revisada de la Pérdida de Suelo" (RUSLE), usan IVs como insumo para estimar uno de los factores que lo componen: el factor vegetal (C). En su trabajo utiliza una técnica basada en inteligencia artificial (Programación Genética), para ello planteó su problema como una optimización, donde el objetivo fue encontrar el índice de vegetación que muestre una mejor correlación con datos de campo del Factor C. Entonces desarrolló un algoritmo basado en GP, el cual construye nuevos índices de vegetación mediante la recombinación iterativa de un conjunto de operadores numéricos y bandas espectrales, pertenecientes a imágenes *Landsat* [19].

En el área de la medicina, el departamento de Bioinformática de la Universidad de Ciencias Informáticas de Cuba, desarrolló una plataforma para el modelado virtual de compuestos orgánicos. El sistema consta con un módulo para el cálculo de descriptores topológicos y topográficos, un visualizador y editor de estructuras químicas, un lenguaje descriptor y varios módulos de Inteligencia Artificial que se integraron para el desarrollo de modelos SAR (*Structure Activity Relationship*) y QSAR (*Quantitative Structure Activity Relationship*), que incluyen Programación Genética, Árboles de Regresión, Lógica Difusa y

Máquinas de Soporte Vectorial. Esos módulos fueron capaces de predecir la actividad biológica en diferentes compuestos orgánicos. El trabajo consistió en desarrollar los *plug-ins* para los módulos que utilizaron la GP y Arboles de Regresión, los cuales sólo se reducían a un conjunto de clases agrupadas en librerías independientes de la plataforma. Esos *plug-ins* se incorporaron al *Front-End* de Grato, permitiendo al usuario generar modelos y predecir actividad biológica en diferentes moléculas [20].

Siguiendo en el área de la medicina, se encontró otro estudio donde se emplea un GP en el contexto de predicción para generar automáticamente una ecuación matemática que represente lo más cercano posible la curva de crecimiento infantil (referentes a peso y altura) registrada históricamente. El método propuesto está enfocado en la inducción de ecuaciones matemáticas, ejecutando el proceso llamado regresión simbólica sobre datos de cuatro criaturas de sexo masculino. En el resultado se observó que la curva generada, se encuentra próxima a la curva de crecimiento de los últimos puntos de la fase de entrenamiento. Así, cuanto mayor será el número de datos para el entrenamiento, mayor será la precisión de la predicción, según los autores. Concluyen que el estudio propuesto retrata tendencias de evolución con base en informaciones del histórico de los niños analizados, y puede ser utilizada para identificar precozmente posibles problemas de desnutrición, obesidad y otros aspectos relacionados al desenvolvimiento físico, sirviendo de auxilio en la toma de decisiones por parte del especialista del área de la salud [21].

En Ingeniería Civil, el estudio fue realizado con el objetivo principal de mostrar las virtudes del empleo de GP como herramienta en la extracción de conocimiento de datos experimentales. Se presentó un sistema de GP distribuida, que aplica un algoritmo de *clustering* para alcanzar soluciones más heterogéneas. La finalidad del sistema fue extraer fórmulas matemáticas (regresión simbólica) a través de un conjunto de datos de entrada y sus correspondientes salidas utilizando para ello una versión distribuida del método de GP clásico. La distribución del algoritmo permite disminuir considerablemente el número de pruebas necesarias para alcanzar una solución óptima, disminuyendo a su vez el coste humano y de recursos asociados [22].

Este artículo [23] ilustra el uso de algoritmos evolutivos como técnica de minería de datos para el descubrimiento de reglas de predicción en bases de datos, reglas que se utilizarán en la mejora de Cursos Hipermedia Adaptativos basados en Web. La idea consiste en descubrir relaciones importantes entre los datos de utilización recogidos durante las

ejecuciones de los distintos usuarios. Esta información puede ser de gran utilidad para el creador del curso, que puede decidir qué modificaciones son las más adecuadas para mejorar el rendimiento de los alumnos. Para la realización de la búsqueda de reglas de predicción se ha utilizado Programación Genética Basada en Gramáticas (GBGP) con técnicas de optimización multiobjetivo. El trabajo también presenta la herramienta gráfica de descubrimiento de reglas que se ha desarrollado para facilitar la utilización de la metodología propuesta de mejora de cursos.

Observando este trabajo [24], el mismo presenta un enfoque de programación genética para la generación de prototipos de clasificación, basados en una metodología de reconocimiento de patrones en el que el conjunto de entrenamiento de un problema de clasificación está representado por un pequeño subconjunto de casos. La asignación de etiquetas para probar casos se hace generalmente por una regla 1NN. Ellos proponen un nuevo método de generación de prototipo, basado en la programación genética, en la que los ejemplos de cada clase se combinan automáticamente para generar prototipos de clasificación altamente eficaces. El programa genético maximiza una estimación de la generalización de rendimiento de un clasificador 1NN usando los prototipos. Afirman que los resultados experimentales mostraron la validez de su enfoque, ya que el método propuesto supera a la mayor parte del estado de las técnicas cuando se utilizan conjuntos de datos grandes y pequeños. Afirman que se obtienen mejores resultados para los conjuntos de datos con sólo atributos numéricos, aunque el rendimiento de su método en los datos mixtos es muy competitivo también.

La previsión del nivel de agua en varios intervalos de tiempo utilizando los registros de las series de tiempo es de importancia en la ingeniería y la gestión de los recursos hídricos. En los últimos 20 años, los enfoques emergentes sobre las técnicas de análisis armónicas convencionales se basan en el uso de programación genética (GP) y Redes Neuronales Artificiales (RNA). En el presente estudio, el GP se utiliza para predecir las variaciones del nivel del mar, tres pasos de tiempo por delante, para un conjunto de intervalos de tiempo que comprende 12 h, 24 h, 5 días y 10 días de intervalos de tiempo utilizando los niveles del mar observados. Las mediciones de un solo medidor de mareas en *Hillarys Boat Harbor*, Australia Occidental, se utilizaron para entrenar y validar el GP empleado para el período comprendido entre diciembre de 1991 y diciembre de 2002 los parámetros estadísticos, es decir, la raíz del error cuadrático medio, coeficiente de correlación y el índice de dispersión, se utilizan para medir sus actuaciones. Estos se compararon con un conjunto correspondiente de resultados

publicados utilizando un modelo de red neuronal artificial. Los resultados muestran que ambas metodologías de inteligencia artificial funcionan satisfactoriamente y pueden ser considerados como alternativas al análisis armónico [25].

Otro estudio similar al anterior trata sobre las variaciones de los niveles de agua de mar que son un motivo de preocupación para los destinos internacionales y costeras que tienen aguas poco profundas. La seguridad en las actividades marítimas, así como la vida humana en esos lugares se puede garantizar mediante el uso de pronósticos con precisión de los niveles de agua. El análisis armónico se emplea tradicionalmente para predicciones de marea, pero a menudo los valores de las predicciones no son idénticos. La diferencia se denomina anomalía del nivel de mar, esto se puede atribuir a la no inclusión de parámetros meteorológicos como insumo para la predicción de la marea. Por lo tanto otras técnicas de predicción se hacen necesarias. Los estudios anteriores sobre las predicciones del nivel del mar indican una mejor eficiencia de las técnicas alternativas como la red neuronal artificial (ANN) y la programación genética (GP), y otros investigadores han utilizado series temporales. Este trabajo predijo los niveles del mar indirectamente mediante la predicción de anomalías del nivel del mar (SLA) utilizando componentes de la velocidad de vientos locales de la hora actual hasta 12 horas anteriores, como entradas en las cuatro estaciones del año de las costas de EE.UU. con las técnicas GP y ANN. Las medidas de error y los gráficos indican que las predicciones fueron satisfactorias [26].

En [27], afirma que el consumo de energía es un índice económico importante, que refleja el desarrollo industrial de una ciudad o de un país. Pronosticar el consumo de energía por métodos estadísticos convencionales por lo general requiere la realización de supuestos tales como la distribución normal de los datos de consumo de energía o en una muestra de gran tamaño. Sin embargo, los datos recogidos en el consumo de energía a menudo son muy pocos o no normales. Desde un modelo de pronóstico *Grey*, basado en la Teoría *Grey*, se puede construir por lo menos cuatro puntos de datos o ambigüedad de datos, que puede adoptarse para pronosticar el consumo de energía. En algunos casos, sin embargo, un modelo de pronóstico *Grey* puede producir grandes errores de predicción. Para minimizar este tipo de errores, el estudio desarrollo un modelo de predicción *Grey* mejorado, que combina la modificación residual con una estimación de programación genética. Finalmente, un caso real de consumo de energía en China, se consideró para demostrar la eficacia del modelo de predicción propuesto.

Este trabajo investigo la capacidad de la programación genética (GP) y el sistema de inferencia *neuro-fuzzy* (ANFIS), que son técnicas de adaptación para el pronóstico profundidad de aguas subterráneas. Cinco diferentes modelos de GP y ANFIS que comprenden diversas combinaciones de valores de profundidad del nivel freático a partir de dos estaciones, *Bondville* y *Perry*, se desarrollan para pronosticar una, dos y tres días por delante las profundidades de la capa freática. El error cuadrático (RMSE), el índice de dispersión (SI), cuenta la varianza de (VAF) y el coeficiente de determinación estadístico (R2),se utilizan para evaluar la precisión de los modelos. Sobre la base de las comparaciones, se encontró que los modelos GP y ANFIS podrían ser empleados con éxito en la predicción de las tablas de fluctuaciones de agua profunda. Sin embargo, GP es superior a ANFIS en dar expresiones explícitas para el problema [28].

El pronóstico con precisión del movimiento de contenedores es crucial para el éxito de cualquier política de operación de un puerto. Este estudio creó un modelo predictivo óptimo de volúmenes de movimiento de contenedores en los puertos mediante el uso de programación genética (GP), el enfoque de descomposición (X-11), y la regresión auto estacional integrado de media móvil (SARIMA). Se recogieron veintinueve años de datos históricos de los principales puertos de Taiwan para establecer y validar un modelo de pronóstico. Los niveles de porcentaje de error absoluto entre previsión y datos reales estaban dentro del 4% para los tres enfoques. Las predicciones del modelo GP iban 32-36% mejor que los de X-11 y SARIMA. Estos resultados sugirieron que GP es el método óptimo para este caso [29].

Este artículo [30], presenta un enfoque de la programación genética (GP) para predecir los coeficientes de dispersión longitudinal en los arroyos naturales. Los datos publicados fueron compilados de la literatura sobre el coeficiente de dispersión para una amplia gama de condiciones de flujo, y se utilizaron para el desarrollo y pruebas del método propuesto. El enfoque propuesto GP produjo excelentes resultados (R2 = 0,98 y RMSE = 0,085) en comparación con los predictores existentes (*Rajeev* y *Dutta*, *Hydrol* Res 40 (6): 544-552, 2009, R2 = 0,345 y RMSE = 1778,6) para el coeficiente de dispersión.

Con el crecimiento de la *Linked Data Web*, la relación tiempo-eficiencia para el cálculo de los vínculos entre las fuentes de datos se han convertido en indispensable. En este trabajo [31], se ha presentado EAGLE, que es un enfoque de aprendizaje activo basado en la programación genética. EAGLE genera especificaciones de enlace de alta precisión que

reduce el tiempo de la carga de anotación para el usuario. Evaluaron EAGLE contra el aprendizaje por lotes en tres conjuntos de datos diferentes y demostraron que su algoritmo puede detectar especificaciones con un *F-measure* superior a 90% mientras que requiere un pequeño número de preguntas.

2.7 REPRESENTACIÓN DE LOS PROGRAMAS

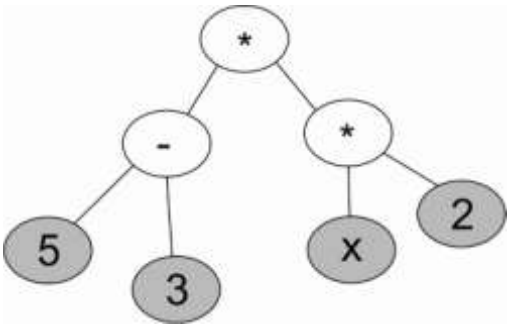
La programación genética en su codificación desarrolla programas de computadores representados en la memoria como estructuras de árbol [8]. Los árboles pueden ser fácilmente evaluados en forma recursiva.

En el árbol se puede identificar dos tipos de nodos:

Terminales u hojas del árbol: no tienen hijos, se asocian con valores constantes o variables.

El conjunto de símbolos terminales (*Figura 2-2*) está formado por los siguientes elementos:

- Los valores de entrada o input.
- Constantes utilizadas por el programa que pueden ser del tipo entero, real, booleano, etc.
- Funciones sin argumentos (Ej. Función de números aleatorios: rand())

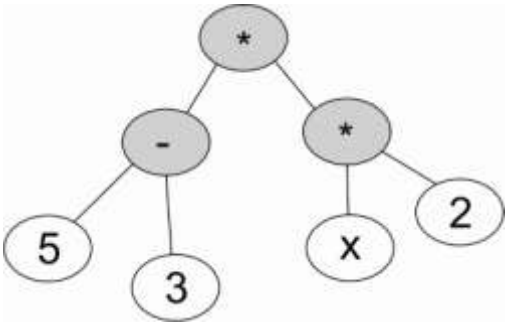


f(x) = (5-3)*(x*2)

Figura 2-2 - Nodos terminales (5),(3),(x)y(2) [8]

Funciones: que tienen uno o más hijos y se asocian con operadores del algoritmo a desarrollar (*Figura 2-3*). Ejemplos de este conjunto son las siguientes funciones:

- Aritméticas: suma, resta, multiplicación y división.
- Trigonométricas y Logarítmicas: seno, coseno, ln, exp, etc.



f(x) = (5-3)*(x*2)

Figura 2-3 - Nodos funcionales (-),(*)y(*) [8]

En la GP es importante definir al inicio el conjunto de funciones y terminales, donde con cada nodo bien especificado del algoritmo procederá a construir los árboles. Se debe configurar el algoritmo para que sepa qué operadores utilizar. En consecuencia, se debe ajustar el número de operadores a los necesarios.

Al conjunto de terminales y funciones, hay que agregarles dos requisitos, cerradura y suficiencia [8]. La suficiencia dice al algoritmo que la solución del problema puede ser especificada con el conjunto de operadores asignados y la cerradura establece la longitud de los árboles.

2.8 DIAGRAMA DE FLUJO DE LA PROGRAMACIÓN GENÉTICA

El algoritmo del GP es similar al de los algoritmos genéticos, se basa en la generación de sucesivas de poblaciones, pero en este caso de programas con estructura de árbol, creada a partir de una población inicial generada aleatoriamente. Esto se demuestra en el diagrama de flujo que sigue a continuación en la *Figura 2-4*.

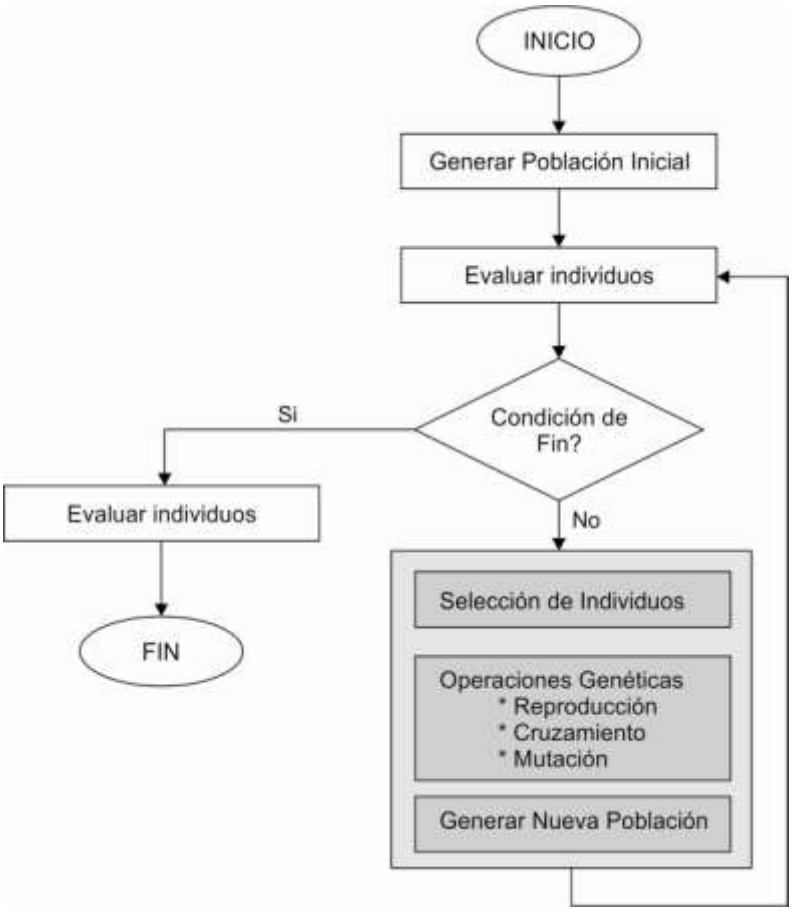


Figura 2-4–Diagrama de flujo para un programa genético [8]

La población inicial se crea de forma aleatoria y posteriormente se forman las sucesivas generaciones, con las operaciones de cruces, reproducciones y mutaciones de los individuos de cada generación anterior. La evaluación de los programas se encuentra a cargo del *fitness*, a los efectos de poder identificar al programa que logre el mejor resultado del problema propuesto.

2.9 GENERACIÓN INICIAL

Cada árbol en el primer paso del algoritmo se creará de manera aleatoria, teniendo en cuenta las restricciones que existen en los árboles. Para la creación de un árbol existe una gran variedad de algoritmos y los más utilizados son: parcial, completo e intermedio [8].

El algoritmo de creación parcial genera árboles donde la altura máxima no supera la especificada, con este algoritmo cada hoja tendrá como máximo la profundidad especificada. El algoritmo completo, genera árboles cuyas hojas están todas a un determinado nivel establecido como máximo. En el algoritmo de creación intermedio, se mezclan los dos

anteriores, y fue creado para que exista una mayor variedad en la población inicial, y así lograr una mayor diversidad genética.

Con estos algoritmos se evita que se generen árboles de altura 1, o sea, árboles que contengan sólo un elemento terminal, y en la práctica se modifican los algoritmos para evitar esa posibilidad [8].

2.10 OPERADORES GENÉTICOS

La programación genética funciona de manera similar a los algoritmos genéticos, después de haber creado la población inicial, se pasa al proceso evolutivo, donde a partir de combinaciones de individuos de la población, se van formando los nuevos que dan lugar a la nueva población. Los individuos son seleccionados de una forma probabilística, dando lugar a que los individuos más aptos sean seleccionados el mayor número de veces. En consecuencia los operadores genéticos son exactamente iguales al de los algoritmos genéticos. Sin embargo por la forma diferente de codificación, es necesario modificar de cierta forma los operadores para adaptarlos a la programación genética.

2.10.1 CRUCE

El cruce es el operador principal, donde dos individuos de la población son seleccionados y se combinan para crear otros individuos nuevos. Después de seleccionar los dos padres, se selecciona un nodo al azar en el primero y otro en el segundo para realizar el intercambio, este cruce se efectúa mediante el intercambio de los subárboles seleccionados, como se muestra en la *Figura 2-5*.

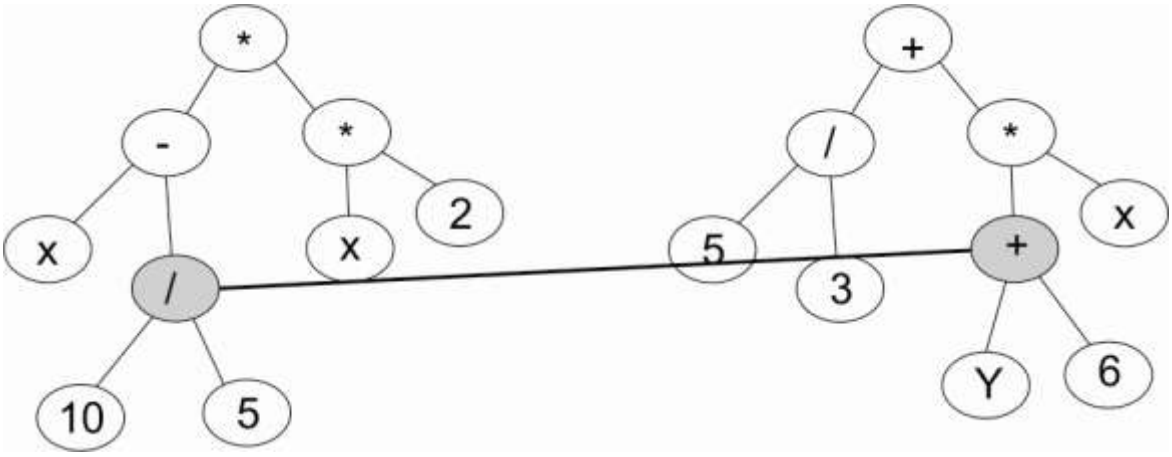


Figura 2-5 - Individuos y nodos seleccionados para el cruce [8]

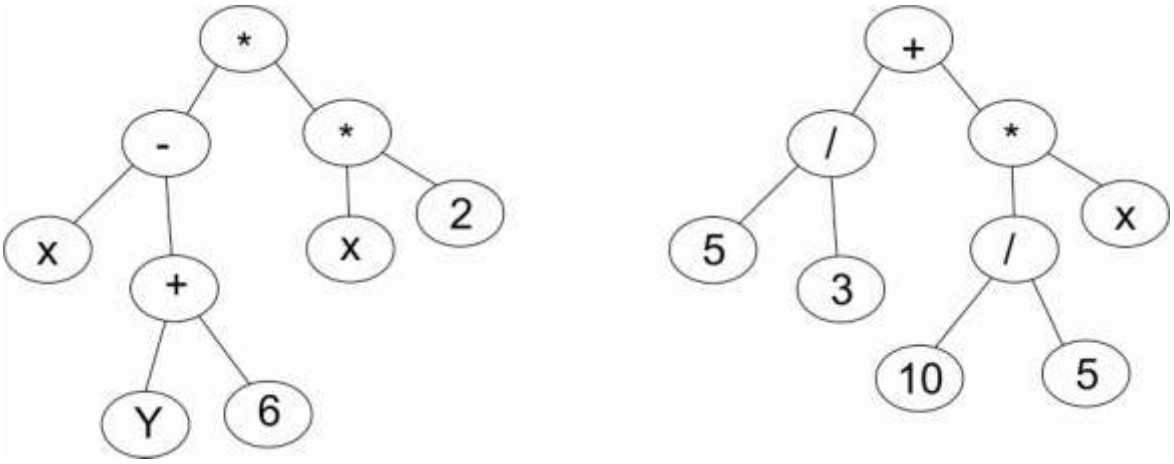


Figura 2-6 - Resultado del cruce [8]

Finalmente se introducen los árboles nuevos en la nueva generación, y en este caso el cruce se lo relaciona a un método sexual, ya que se necesitan de dos individuos para generar a los individuos nuevos.

2.10.2 MUTACIÓN

El operador de mutación realiza una variación de un árbol en una población, y para que el mismo se ejecute, se le asigna una probabilidad que generalmente es baja. Se pueden realizar de dos formas, mutación para variar un solo nodo, o mutación para variar una rama entera del árbol.

Primer tipo, mutación puntual actúa de la siguiente manera:

1. Seleccionar un nodo al azar del árbol.
2. Se selecciona al azar un nodo del conjunto de terminales o funciones, del mismo tipo que el seleccionado en el individuo.
3. Se reemplaza el nodo antiguo por el nuevo.

En el segundo tipo de mutación, los cambios son mayores en el árbol. Se realiza de la siguiente manera, se escoge un nodo al azar, se elimina todo el subárbol que cuelga del nodo, y se crea un nuevo subárbol del tipo y altura, y se reemplaza en su lugar.

2.11 EVALUACIÓN

Como ocurre en el algoritmo genético, la valoración del *fitness* de un individuo se realiza por medio de su ajuste [7]. Este valor representa lo bien que el árbol soluciona el problema. En consecuencia la programación genética utiliza el mismo método de seleccionar cuales individuos serán los afectados por los operadores genéticos de reproducción, cruce y mutación, en base a esta métrica denominada función de aptitud o *fitness*.

El valor de aptitud tiene dos fines, una de medir que tan buena es la solución que genera cada individuo para su proceso de evolución y la segunda para elegir qué individuo sería el que mejor resuelva el problema en cuestión.

2.12 PARÁMETROS

La programación genética tiene un conjunto de parámetros que deben ser configurados para su ejecución.

- Tamaño de la población: se refiere al número de individuos que puede tener la población. Se debe de tratar de ajustar este número de forma proporcional a la complejidad del problema, tomando valores altos cuanto mayor sea ésta.
- Altura máxima del árbol: indica la altura máxima que podrán tener los árboles en el proceso evolutivo.
- Altura máxima de los árboles iniciales.
- Altura máxima que tendrán los subárboles: en el caso de las mutaciones que afecten a los subárboles, hay que asignar un tamaño máximo para su mutación y que se ajuste a la altura máxima del árbol principal.
- La tasa de cruces: sería el porcentaje de individuos de la siguiente generación que serán creados a partir de cruces de individuos de la anterior generación.
- La probabilidad de mutación: esta debe ser muy baja.
- Algoritmos de creación, reproducción y mutación utilizados.
- El criterio de parada del algoritmo, estos pueden ser:
    - Número máximo de generaciones.
    - Alcanzar el valor de ajuste.
    - Convergencia de la población.

2.13 DISCUSIONES Y COMENTARIOS

Como se vio en este apartado, los algoritmos genéticos son un avance y logro de la inteligencia artificial, ya que utiliza el comportamiento biológico sobre un proceso automatizado de programas de computadores. En este caso son algoritmos de búsqueda basados en el mecanismo de la selección natural y la genética.

Los GA en cada generación crean los individuos que son cadenas de bits, involucrando un sistema aleatorio que genera la evolución de los mismos, con métodos absolutamente complejos como se pudo observar. Son complejos porque deben lograr que los hijos hereden material genético de los padres, inclusive pudiendo mutar en algunas de las generaciones, y aunque algunos de estos individuos no generen buenos resultados para el problema, existirán otros que sí irán evolucionando y en consecuencia mejorarán en cada paso la solución a la que se quiere llegar.

El algoritmo genérico muestra el proceso práctico de la teoría mencionada, la cual está orientada a lograr el objetivo principal de todo algoritmo genético, el diseño de soluciones de alta calidad para problemas de elevado grado de complejidad, con la habilidad de adaptar las soluciones por medio de los cambios en el entorno en el que se desarrollan.

La GP tiene un funcionamiento similar al de los algoritmos genéticos, la diferencia se encuentra en este caso sobre el tratamiento de los individuos. Estos son representados como programas de computadores que se codifican en forma de árbol, conteniendo terminales y funciones.

La aplicación de los operadores evolutivos en este método es mucho más compleja, ya que trabaja con individuos mucho más extensos. Entonces por su forma diferente de codificar, se modifican los operadores para que se puedan adaptar a la programación genética.

CAPITULO III
PROGRAMACIÓN GENÉTICA LINEAL

3.1 INTRODUCCIÓN

En este apartado se desarrolló propiamente el estudio en base a lo que es la programación genética lineal (LGP), explicando en base a la teoría sus conceptos y los procedimientos realizados para su aplicación práctica sobre el problema planteado en el primer capítulo de esta tesis.

Los conceptos son los fundamentos esenciales que muestran el camino tomado para el desarrollo de este proyecto. Toda la técnica empleada dependió de lo establecido en la teoría y las aplicaciones realizadas por los estudiosos del área que se investigó.

La LGP no escapa de la línea de los algoritmos evolutivos con la aplicación de la teoría darwiniana. Los individuos a evolucionar son secuencia de instrucciones de programas de computadores elaborados en un lenguaje de programación imperativa y que utilizan una estructura lineal.

La LGP tiene una serie de características especiales y para su aplicación requieren de ciertos pasos que se necesitan cumplir para su elaboración. Son complejos a la hora de ajustar y ejecutar, los mismos se encuentran descriptos en esta parte del desarrollo.

La representación de los individuos es esencial para entender la metodología, ya que son conjunto de registros que deben almacenar los valores generados en cada una de sus líneas que pueden ser ejecutadas o no dependiendo de la efectividad del código generado.

Los operadores evolutivos tienen el mismo objetivo como en los algoritmos genéticos o la programación genética en sí, la diferencia se encuentra en la forma de aplicar, ya que la estructura difiere mucho de los individuos tratados en las otras metodologías.

El desarrollo de la aplicación de esta metodología se verá a continuación, los cuales se encuentran explicados con los datos establecidos en el problema planteado y utilizados para llegar a la solución del problema en cuestión, la predicción de ingresos de causas penales mediante la programación genética lineal.

3.2 ESTADO DEL ARTE LGP

En este material [32],se presentó a la Programación Genética Lineal (LGP), que es una extensión de GP, como una herramienta alternativa en la predicción de profundidad para tuberías sumergidas. Los conjuntos de datos de las mediciones de laboratorio se obtuvieron de la literatura publicada y se utilizaron para desarrollar modelos LGP. El modelo propuesto LGP se comparó con sistemas adaptativos de inferencia *neuro-fuzzy* (ANFIS). Se observó que las predicciones de la LGP dan un buen resultado con los datos medidos, y bastante mejor que la ANFIS y la ecuación de regresión.

Los datos climáticos diarios, la temperatura del aire, radiación solar, velocidad del viento, presión y humedad de las tres estaciones meteorológicas automáticas de Fresno, Los Ángeles y San Diego, en California, se utilizan como datos para la LGP para estimar la evaporación. Las estimaciones LGP se compararon con los de la programación de expresión génica (GEP), que es otra rama de la GP, perceptrones multicapa (MLP), redes neuronales de base radial (RBNN), regresión generalizada redes neuronales (GRNN) y *Stephens-Stewart*. Las actuaciones de los modelos se evaluaron utilizando raíz media de errores cuadrados (RMSE), error absoluto medio (MAE) y el coeficiente estadístico de determinación (R2). Sobre la base de las comparaciones, se encontró que la técnica LGP podría emplear con éxito en el modelado proceso de evaporación a partir de los datos climáticos disponibles [33].

En el trabajo [34], se vio modelos basados en LGP que se construyeron utilizando dos conjuntos diferentes de datos de entrada. El primer conjunto de entradas comprende diámetro de cilindro de hormigón, la resistencia del hormigón no confinado, resistencia a la tracción de laminado CFRP y el espesor total de las capas de CFRP utilizadas. El segundo grupo incluye la resistencia del concreto no confinado y la presión de confinamiento final que son los parámetros más utilizados en el confinamiento CFRP de los modelos existentes. Se desarrollaron los modelos basados en resultados experimentales obtenidos de la literatura disponible. Los resultados demuestran que las fórmulas a base de LGP son capaces de predecir la resistencia final y la compresión de cilindros de hormigón con un nivel aceptable de precisión. Los resultados LGP también se compararon con varios modelos de confinamiento CFRP presentados en la literatura y se han encontrado para ser más exactos, en casi todos los casos. Por otra parte, las fórmulas desarrolladas por LGP son bastante cortas y sencillas y prácticas para su utilización.

Se observó este estudio del mercado de comercio de divisas (*forex*), donde utilizaron un sistema de programación genética lineal (LGP) para el mercado automatizado de *Forex* sobre cuatro pares de divisas principales. Se consideraron funciones de *fitness* con diferentes grados de conservadurismo a través de la incorporación de la máxima pérdida. El uso de los tipos de acondicionamiento físico en el sistema de LGP de diferentes tendencias de valor de moneda se examinaron en términos de rendimiento en el tiempo, que subyace a las estrategias comerciales y la rentabilidad global. Con un análisis de la rentabilidad del comercio mostraron que el sistema de LGP es muy preciso, tanto en la compra para lograr ganancias y venta para evitar pérdidas, con niveles moderados de actividad comercial [35].

La estimación correcta de la concentración de sedimentos en suspensión transportada por un río es muy importante para muchos proyectos de recursos hídricos. En este trabajo [36], se propuso la aplicación de la programación genética lineal (LGP) para la estimación de la concentración de sedimento en suspensión. La LGP se comparó con los de las redes neurales adaptativas, *neuro-fuzzy* y modelos de curvas de calificación. El caudal diario y datos de concentración de sedimentos en suspensión a partir de dos estaciones, la estación de Río Valenciano y la estación de Quebrada Blanca, operado por el *US Geological Survey* (USGS) se utilizaron como estudios de caso. La raíz media de los errores cuadrados (RMSE) y el coeficiente estadístico de determinación (R2), se utilizaron para evaluar la exactitud de los modelos. La comparación de los resultados indicó que la LGP se comporta mejor que las redes neuronales, *neuro-fuzzy* y modelos de curvas de calificación.

Surcos laterales son estructuras de derivación utilizados ampliamente en el riego, la protección contra inundaciones y sistemas de alcantarillado combinado son necesarios. La estimación precisa del coeficiente de descarga (Cd) de los vertederos laterales es esencial para calcular el perfil de la superficie del agua a través de estos y para determinar la tasa de flujo de salida lateral del sistema. En este trabajo, se utilizó una técnica de programación genética lineal (LGP) para desarrollar nuevas fórmulas empíricas para la estimación de la Cd de los vertederos laterales rectangulares con aristas ubicadas en canales circulares. Han empleado un total de 1.686 observaciones experimentales de laboratorio en ambos regímenes de sub y super-flujo críticos con el fin de capacitar y validar los modelos propuestos. El rendimiento de los modelos basados en LGP se compararon con los de diferentes modelos de regresión lineal múltiple. Para determinar la eficiencia de los modelos se utilizaron la raíz media de los errores cuadrados, el error absoluto medio y el coeficiente de determinación. Los resultados indicaron en forma explícita, que el modelo basado en LGP utilizando funciones matemáticas podrían

emplear con éxito en la estimación de Cd en ambas condiciones de flujo suby super-críticos[37].

En este trabajo [38], el autor manifiesta que un número de investigadores han tratado de tener sistemas de comercio GP exitosas, haciéndolas mejores mediante el uso de filtros. Investigó el uso de un sistema de programación genética lineal (LGP) que combina las señales de GP prestados a través de múltiples marcos de tiempo para producir una acción comercial. Examinó cuatro combinaciones de marcos de tiempo que se extienden en el pasado. También se examinó dos diferentes mecanismos de toma de decisiones para evaluar el conjunto de señales GP con los marcos de tiempo, uno basado en el voto de la mayoría y otro basado en la proximidad temporal de la decisión de compra. Los resultados indicaron con énfasis que el voto mayoritario superó a la proximidad de los plazos para la decisión de negociación en curso. Los análisis también indicaron que las combinaciones más largas de marco de tiempo eran más conservadores y superaron a las combinaciones más cortas para ambas tendencias generales de precios en la alza como en la baja.

En este estudio [39], las técnicas de programación basada en árbol genético (TGP) y sus variantes recientes, programación genética lineal (LGP) y la programación genética de la expresión (GEP), se utilizan para desarrollar nuevas ecuaciones de predicción para la capacidad de elevación de las cámaras de succión. La capacidad de elevación se formula en términos de variables flexibles. Una base de datos experimentales obtenidos a partir de la literatura se empleó para desarrollar los modelos. Además, un análisis estadístico convencional se realizó para referencia de los modelos propuestos. El análisis de sensibilidad y de parámetros se llevó a cabo para verificar los resultados. TGP, LGP y GEP fueron comprobados para ser métodos eficaces para evaluar la capacidad de elevación horizontal, vertical e inclinado de cámaras a succión. Los modelos de TGP, LGP y GEP alcanzaron un rendimiento de predicción mucho mejor comparando con los modelos que se encuentran en la literatura.

En este trabajo [40], se presenta la programación lineal genética (LGP), y un algoritmo de búsqueda híbrido del LGP y el recocido *simulated annealing* (SA), llamado LGP/SA, para predecir las características de rendimiento de suelo estabilizado. LGP y LGP/SA relacionaron la resistencia a la compresión no confinada (UCS), densidad máxima de la seca (MDD), y el contenido óptimo de humedad (OMC), métricas de suelo estabilizado a las propiedades del suelo natural, así como los tipos y cantidades de aditivos estabilizantes. Diferentes conjuntos

de modelos de predicción basados en LGP y LGP/SA, han sido desarrollados por separado. Las contribuciones de los parámetros que afectan a UCS, MDD, y OMC fueron evaluados a través de un análisis de sensibilidad. Un análisis de parámetros posterior se llevó a cabo y las tendencias de los resultados se compararon con los estudios previos. Un completo conjunto de datos obtenidos de la literatura se ha utilizado para el desarrollo de los modelos. Los resultados experimentales confirman que la exactitud de los modelos propuestos es satisfactoria. En particular, los modelos basados en LGP encontraron valores más precisos que los modelos basadosen LGP/SA.

En este artículo, la programación genética lineal (LGP) se utiliza para predecir la radiación solar global. La radiación solar se formula en términos de varios parámetros climatológicos y meteorológicos. Bases de datos completas que contienen datos mensuales recogidos durante 6 años (1995-2000) en dos ciudades nominales en Irán, se utilizaron para desarrollar los modelos basados en LGP. Se establecen modelos separados para cada ciudad. Para verificar el funcionamiento de los modelos propuestos, se aplican para estimar la radiación solar global de los datos de prueba de la base de datos. La contribución de los parámetros que afectan a la radiación solar, se evaluaron a través de un análisis de sensibilidad. Los resultados indicaron que los modelos LGP dan estimaciones precisas de la radiación solar global y superan significativamente el modelo tradicional de *Angstrom* [41].

Este estudio [42], se presenta variantes prometedoras de la programación genética (GP), la programación genética lineal (LGP) y la programación multi-expresión (MEP) para evaluar la resistencia a la licuación de suelos arenosos. Generalizando LGP y las relaciones basadas en MEP, se desarrollaron entre la densidad de energía de deformación necesaria para disparar la licuación (la capacidad dela energía) y los factores que afectan a las características de licuación de arenas. Las correlaciones se establecieron en base a los resultados experimentales obtenidos a partir de la literatura. La validación externa de los modelos se verificó usando criterios estadísticos recomendados por los investigadores. La sensibilidad y los análisis de parámetros se realizaron para la verificación de las correlaciones. Los resultados indican que las correlaciones propuestas son efectivamente capaces de capturar la resistencia de licuación de un número de suelos arenosos. Las correlaciones desarrolladas proporcionan un rendimiento significativamente de mejor predicción que los modelos encontrados en la literatura. Además, los mejores modelos LGP y MEP obtuvieron un resultado superior que el modelo óptimo GP tradicional.

Viendo dos variantes de la programación genética, la programación genética lineal (LGP) y la programación multi-expresión (MEP), se utilizaron para detectar episodios de la fibrilación auricular (FA). Modelos basados en LGP y MEP, se derivaron para clasificar muestras de FA y episodios normales basados en el análisis de señales de intervalo RR. Un análisis de regresión por mínimos cuadrados ponderados (WLS) se realizó utilizando las mismas características y conjuntos de datos de referencia. Los modelos se han desarrollado utilizando la base de datos de arritmias del MIT-BIH. Las actuaciones de diagnóstico de los clasificadores LGP y MEP fueron evaluados a través de las características de funcionamiento del receptor (ROC). Los resultados indican que los modelos LGP y MEP son capaces de diagnosticar la arritmia AF con una alta precisión. Los modelos propuestos tienen significativamente mejores actuaciones de diagnóstico que la regresión y varios modelos encontrados en la literatura [43].

3.3 CONCEPTOS BÁSICOS DE LA LGP

LGP es una variación de la Programación Genética (GP), que opera secuencia de instrucciones imperativas de un lenguaje de programas de computadoras, incluyendo diversas formas de ejecución del programa [1]. En consecuencia la LGP es una evolución de la GP, donde el mayor cambio ocurre en la estructura de los programas. El tratamiento de estos programas en la GP es a través de estructura de árboles, sin embargo en la LGP los programas utilizan una estructura lineal.

La LGP se ha probado e interpretado generalmente en el lenguaje de alto nivel C, sin embargo, está comprobado que los conceptos de la programación genética puede ser traducida en la mayoría de los lenguajes de programación imperativos modernos [10], e inclusive a nivel de lenguaje de máquina. En esta tesis se ha realizado el algoritmo en el lenguaje de programación JAVA.

LGP es un tipo de Algoritmo Evolutivo, donde los cromosomas son estructuras de datos que sufren la adaptación, y como ya se ha dicho, éstos en sí son programas de computador [46]. Utilizan operadores genéticos especiales que generalizan la combinación sexual y la mutación, para los programas de computador estructurados en forma lineal que están bajo adaptación.

3.4 CARACTERÍSTICAS DE LA LGP

- Material genético lineal. El material genético lineal es una de las reglas en los Algoritmos Genéticos, y en este caso viene acompañado en una estructura lineal de programas de computador.
- Material genético de longitud variable. La LGP trabaja sobre material genético que puede variar de tamaño. Pero por razones prácticas, generalmente se implementan limitaciones en el crecimiento, trabajando libremente podría permitirse crecimientos considerables a partir de una generación original que se produce aleatoriamente.
- Material genético ejecutable. La LGP trata con la evolución directa de programas de computador. En la mayoría de los casos el material genético que está evolucionando es ejecutable, esto quiere decir que las estructuras son interpretadas por un lenguaje de computación existente. En todos los casos hay un proceso de ejecución del material genético, con el fin de ver directamente el comportamiento de la función deseada, a partir de la cual se obtiene el valor de aptitud.
- Cruce que preserva la sintaxis. En la mayor parte de los casos, se definen los operadores de cruces, de manera que estos preserven la corrección sintáctica del programa que es el material genético, todo esto definido por el lenguaje que se escoge para su representación.

El espacio de búsqueda en la LGP, es el de todos los posibles programas de computador compuestos de funciones y registros apropiados al dominio del problema. Las funciones pueden ser operaciones aritméticas, operaciones de programación, funciones matemáticas, funciones lógicas, etc.

3.5 PASOS APLICADOS PARA LA ELABORACIÓN DEL LGP.

1. Conjunto de registros. Consiste en identificar el conjunto de registros que corresponden a variables o valores constantes, se pueden ver cómo las entradas al programa. El conjunto de registros (junto con el conjunto de funciones) son los componentes a partir de los cuales la LGP, trata de construir un programa de computador para solucionar el problema.

Registros	Descripción
r[0]	Registro de Salida
r[13]	Valor anterior (x_i -1)
r[12]	x_i– 2
r[11]	x_i– 3
r[10]	x_i– 4
r[15]	Número de predicción – i
r[14]	Pronóstico anterior
r[0]-r[9]	Registros variables, que se inicializan con valor 1
r[10]-r[21]	Registros constantes
r[16]	π
r[17]	e
r[18]	Valores aleatorios con rango [-1 a 1]
r[19]	Valores aleatorios con rango [-10a 10]
r[20]	Valores aleatorios con rango [-100a 100]
r[21]	Valores aleatorios con rango [0a 1]

Tabla 3-1 - Registro definidos para la LGP [Fuente propia]

El número de registros totales se debe definir al inicio del programa, y generalmente para todos los problemas no debe ser muy extenso. Como máximo se utiliza la cantidad de 256 registros [1]. Para el problema se ha optado por utilizar la cantidad de 22 registros (*Tabla 3-1*).

La entrada del programa es una sola y se almacena en el r[1] y la salida en el registro se almacena en el r[0], si las entradas son más de una, se utilizan los registros adyacentes.

2. Conjunto de funciones establecidas. Consiste en identificar el conjunto de funciones que se van a usar, para generar la expresión matemática que trata de satisfacer la muestra dada finita de datos. A cada función le corresponde un número de parámetros y un tipo de datos. Cada programa de computador es una composición de funciones y registros, obtenidos de sus respectivos conjuntos.

   Como en la Programación Genética, en la LGP se deben definir las terminales y las operaciones. Las terminales son los valores enteros, reales, las entradas y otros introducidos por el usuario. Las operaciones o funciones son aritméticas, exponenciales, condicionales, trigonométricas y boleanas. (*Tabla 3-2*)

Tipo de Operación	Notación General	Rango Input
Aritméticas	$r_i = r_j + r_k$ $r_i = r_j - r_k$ $r_i = r_j * r_k$ $r_i = r_j / r_k$	$r_i, r_j, r_k \in R$
Exponenciales	$r_i = r_j \wedge r_k$ $r_i = \log(r_j)$ $r_i = \lvert r_j \rvert$	$r_i, r_j, r_k \in R$
Trigonométricas	$r_i = \text{seno}(r_j)$ $r_i = \cos(r_j)$	$r_i, r_j \in R$
Booleanas	$r_i = r_j$ and $r_k r_i$ = or r_k	$r_i, r_j, r_k \in \beta$
Condicionales	If ($r_j <= r_k$) if ($r_j >= r_k$)	$R_i, r_j \in R$

Tabla 3-2 - Operaciones para la LGP [1]

3. Valor de aptitud o *fitness* [8]. Cada programa de computador es un individuo en la población, y se le debe medir en términos, qué tan bien se comporta en el ambiente del problema particular, a esta medida se le conoce como medida de aptitud y la naturaleza de la medida de aptitud varía con el problema. La aptitud de un programa se mide ponderando el error que se produce al ejecutar ese programa, entonces el mejor programa de computador tendrá un error cercano a cero. Los individuos de la población inicial o generación 0, generalmente tienen un valor de aptitud muy baja. Sin embargo, algunos programas de la población serán más aptos que otros, estas diferencias en el comportamiento son las que se deben explotar y explorar seguidamente. El *fitness* del individuo es calculado como inversamente proporcional a una función de error sobre un conjunto de valores de entrada-salida, que es conocido como secuencia de entrenamiento.

   La función de error más utilizada para problemas de aproximación es la suma de los errores cuadrados. Para el proyecto fue utilizada esta ecuación para calcular el *fitness* de cada individuo, y se definió como la suma de los errores cuadrados para "N" valores de la muestra.

$$\mathbf{e = \frac{1}{N} \sum_{i=0}^{N} (x_i - p_i)^2} \qquad \textbf{(3-1)}$$

   Donde "x_i" es el valor real de la posición "i" y el "p_i" es el pronóstico para el periodo "i"

4. Parámetros para controlar la ejecución del algoritmo [1]. Al efectuar las operaciones genéticas sobre la población con la que se trabaja en el momento, la población de hijos

reemplaza la generación anterior, a cada individuo de la nueva población se le mide su aptitud, y este proceso se debe repetir por muchas generaciones. En consecuencia, se deben definir los parámetros que controlarán la ejecución del LGP y estos son: el tamaño de la población; el número de generaciones; las probabilidades de aplicación de los operadores genéticos de cruce y mutación; y la estrategia de selección de los individuos para la nueva generación. Ver *Tabla 3-3*.

Parámetros	Valor
Longitud máxima de cromosomas para los individuos de la población inicial	50
Longitud mínima de cromosomas para los individuos de la población inicial	3
Longitud máxima de cromosomas de los individuos	200
Tamaño de la población	150
Tamaño de la población elite	30
Cantidad de individuos por cada selección	2
Probabilidad de cruzamiento por cada selección	100%
Probabilidad de mutación por cada selección	30%
Probabilidad de intensidad de mutación por cada selección	20%
Cantidad de operadores	9
Cantidad total de registros	22
Cantidad de registros constantes	12
Cantidad de registros randómicos	4
Cantidad de datos para entrenamiento	72
Cantidad de datos para validación del modelo	12

Tabla 3-3 - Parámetros utilizados en la LGP [Fuente propia]

5. El método para designar un resultado y el criterio para terminar la ejecución del programa. El mejor individuo que aparece en cualquier generación de una corrida del algoritmo, es el que se designa como el resultado final.

3.6 REPRESENTACIÓN DE INDIVIDUOS

El concepto de programación imperativa es relacionado estrechamente con el lenguaje de máquina. El lenguaje de máquina indica una secuencia de instrucciones cuyo orden debe ser respetado durante su ejecución. Básicamente una instrucción imperativa incluye una operación en el operando, un registro de origen y una asignación del resultado de esa operación en un registro de destino.

En LGP, un programa genético se describe como una secuencia de longitud variable de instrucciones simples, que representa la solución a un problema dado [50].

Las instrucciones se representan de la siguiente forma:

r[0] = r[1] + r[5]

Donde r[i] equivale a un registro. El individuo es una posible solución al problema y en Programación Genética Lineal son representados como un conjunto de instrucciones el cual debe proporcionar la solución al problema.

Las instrucciones están compuestas por:

- Un registro fuente donde se almacena el resultado.
- Uno o más registros que almacenan los valores con los que se efectuará la operación.
- Una operación del conjunto de operaciones.

Representación de un individuo obtenido:

r[8] = r[2] + r[1]
r[3] = r[6] + r[2]
r[1] = r[5] + r[4]
r[0] = r[1] + r[2]

y el registro r[0] es el registro de salida.

Estos registros almacenan valores, sin embargo se deben crear una serie de registros que son reservados para constantes numéricas y las mismas deben estar protegidas contra escritura.

Los individuos son secuencias de instrucciones y se representan de la siguiente manera **4 – tupla, <op, i, j, k >** donde "op" es la operación a ejecutarse, la "i" es el índice del registro que indica donde se almacenarán los resultados, "j" y "k" son índices de registros donde están asignados los operandos.

Entonces se puede ejemplificar la representación de una instrucción como sigue **<+,0,1,2>** donde aplicando lo dicho anteriormente quedaría de la siguiente forma:

r[0] = r[1] + r[2].

3.7 OPERADORES EVOLUTIVOS

En la programación genética lineal tenemos como operadores evolutivos a la mutación y el cruzamiento, que son los operadores evolutivos tradicionales de la computación evolutiva [8]. Estos operadores se aplican en el LGP pero con algunas diferencias de cómo se ejecutan en la programación genética tradicional.

En LGP se observan dos tipos de variaciones [50], la micro-variación y la macro-variación. La micro-variación trabaja a nivel de una instrucción y puede afectar a un registro, un operador o una constante. Sin embargo, la macro-variación modifica segmentos de instrucciones, que son un conjunto continuo de instrucciones seleccionados de los individuos a ser cruzados, lo cual influye en el tamaño de los individuos o programas. Entonces se puede identificar que la micro-variación es utilizada en el operador evolutivo de mutación. En cambio la macro-variación se aplica en ambos operadores evolutivos, en el cruzamiento se visualiza al seleccionar los segmentos de dos individuos que realizarán el cruce y en la mutación cuando se modifica completamente la instrucción seleccionada del programa.

3.7.1 CRUZAMIENTO LINEAL

En el cruzamiento lineal se seleccionan dos padres de la población, y a partir de los mismos se producirán la misma cantidad de hijos. Cada padre se dividirá en segmentos, donde un segmento equivale a un conjunto continuo de instrucciones del individuo o programa en LGP. Al segmento se le debe designar un tamaño mínimo y uno máximo.

Entonces, se puede definir que los segmentos tengan una cantidad mínima de instrucciones, como por ejemplo una sola instrucción. Y también se debe definir el tamaño máximo del segmento, que no debe exceder la cantidad de instrucciones de un programa. Para el ejemplo que sigue se puede tratar de la siguiente manera:

Programa:

1. **r[9]=r[4]+r[5]**
2. **r[2]=r[6]+r[2]**
3. **r[8]=sen[7]**
4. **r[1]=r[2]+r[7]**
5. **r[0]=r[1]*r[3]**

En este ejemplo se puede establecer la cantidad mínima de segmento en 1 (una) instrucción y la cantidad máxima de 3 (tres) instrucciones. Entonces se pueden representar los segmentos como sigue:

- Segmento 1
  - **r[2]=r[6]+r[2]**
  - **r[8]=sen[7]**
- Segmento 2
  - **r[1]=r[2]+r[7]**
- Segmento 3
  - **r[8]=sen[7]**
  - **r[1]=r[2]+r[7]**
  - **r[0]=r[1]*r[3]**

Si en el caso anterior se establecía que la cantidad máxima es de 7 (siete) instrucciones, el mayor segmento posible tendría 5 (cinco) instrucciones, que se establece por el tamaño total de instrucciones del programa.

Habiendo establecido el tamaño de los segmentos, posteriormente se deben seleccionar dos individuos que serán denominados padres para realizar el cruzamiento. En estos individuos se seleccionarán un punto de cruzamiento, que será donde empieza el segmento a ser intercambiado. Seleccionando los segmentos requeridos en cada padre, se realiza el intercambio de segmentos, que generarán dos nuevos individuos denominados hijos, y estos pasarán a formar parte de la nueva población *Figura 3-1*.

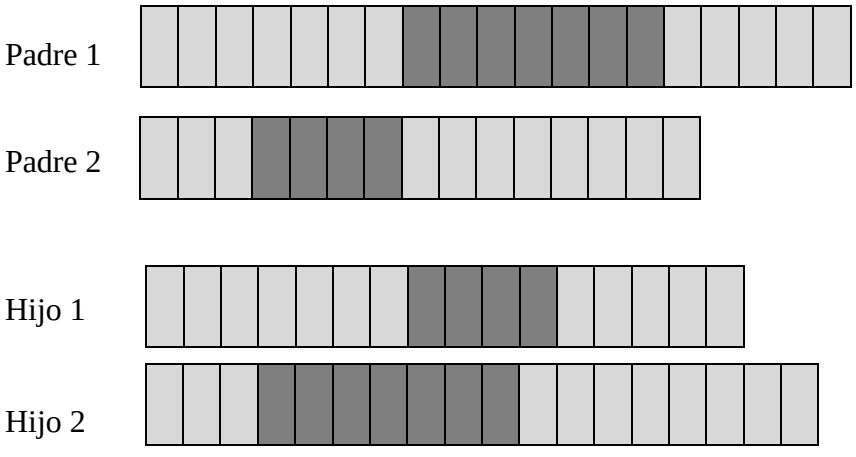


Figura 3-1 - Cruzamiento en Programación Genética Lineal. Figura creada según [1]

Existen casos a tener en cuenta, ya que se tiene definido el tamaño máximo de un individuo en la LGP.Y si el producto de un cruzamiento resulta con un hijo que excede este tamaño, entonces en el intercambio de segmentos se debe realizar el cambio de segmentos idénticos entre los padres para evitar esta situación.

Se había hablado de micro-cruzamiento y macro-cruzamieto, y para ser conciso y concreto, se puede definir un micro-cruzamiento al establecer que el segmento a ser intercambiado sea establecido con una sola instrucción para efectuar la operación. Sin embargo para la macro-mutación el tamaño de los segmentos continuos debe ser establecido como con un número mayor a una instrucción.

Para explicar mejor el cruzamiento se pone como ejemplo los siguientes programas padres de la *Tabla 3-4*:

1. r[2]=r[6]+r[2] **2. r[8]=seno[7]** **3. r[1]=r[2]+r[7]** 4. **r[0]=r[1]*r[3]**	**1. r[7]=r[5]-r[18]** **2. r[7]=r[7]+r[5]** **3. r[1]=r[5]+r[1]** **4. r[2]=seno[7]** **5. r[1]=r[2]+r[2]** 6. **r[0]=r[1]*r[1]**
Padre 1	**Padre 2**

Tabla 3-4 - Segmentos seleccionados para el cruce. Tabla creada según [1]

Los puntos de cruzamiento se definen de la siguiente manera:

- Para el padre 1 desde la instrucción 2, siendo su tamaño de segmento 2(dos) instrucciones.
- Para el padre 2 desde la instrucción 4, y el tamaño de su segmento es de 3(tres) instrucciones.

Al aplicar el operador de cruzamiento entre ambos padres, los hijos que terminan resultando de dicha operación serían los siguientes programas de la *Tabla 3-5*:

1. r[2]=r[6]+r[2] **2. r[2]=seno[7]** **3. r[1]=r[2]+r[2]** **4. r[0]=r[1]*r[1]** **5. r[0]=r[1]*r[3]**	**1. r[7]=r[5]-r[18]** **2. r[7]=r[7]+r[5]** **3. r[1]=r[5]+r[1]** **5. r[8]=seno[7]** **6. r[1]=r[2]+r[7]**
Hijo 1	**Hijo 2**

Tabla 3-5 - Resultado del cruce. Tabla creada según [1]

3.7.2 MUTACIÓN LINEAL

En la mutación lineal, se selecciona una sola instrucción para que se le ejecute este operador evolutivo. Entonces se puede dividir a la mutación en micro y macro mutación la cual se explicara en esta sección [50].

La micro-mutación se realiza al modificar un solo componente de la instrucción seleccionada para mutar, que puede ser los registros o en todo caso el operador. En la macro-mutación sin embargo se inserta o se borra por completo la instrucción. Para el borrado se elige una instrucción al azar y al generar el hijo nuevo, se copia las demás instrucciones del padre, sin la instrucción que fuera seleccionada. Para insertar una instrucción, se crea una instrucción nueva y se inserta en una posición aleatoria en el individuo nuevo, que sería la variación que se le realiza al padre para generar al nuevo individuo.

3.7.2.1 MACRO MUTACIÓN

Algoritmo Macro-mutación

- Un programa en LGP: GP.
- Probabilidad de inserción p_{ins}
- Probabilidad de borrado p_{bor}
- Tamaño máximo del programa t_{max}
- Tamaño mínimo del programa t_{min}

1. Aleatoriamente seleccionar el tipo de macro-mutación (inserción o borrado) de acuerdo a las probabilidades establecidas $p_{ins} \mid p_{bor}$; con $p_{ins} + p_{bor} = 1$
2. Seleccionar de manera aleatoria una instrucción en la posición i (punto de mutación) en el programa GP
3. Si $t(GP)<t_{max}$ y la macro-mutación es inserción

- Crear una instrucción de forma aleatoria
- Insertar la instrucción en la posición i

4. Si t(GP)>t_{min} y la macro-mutación es borrado

   - Eliminar la instrucción en la posición i

Como ejemplo de aplicación del algoritmo se da el siguiente programa en LGP para su entrada:

1. **r[9]=r[4]+r[5]**
2. **r[2]=r[6]+r[2]**
3. **r[8]=sen[7]**
4. **r[1]=r[2]+r[7]**
5. **r[0]=r[1]*r[3]**

Para el punto 1 del algoritmo, se selecciona la inserción como tipo de macro-variación. Para el punto 2 del algoritmo, se selecciona la instrucción 3. Para el punto 3 se genera la nueva instrucción aleatoria (representada por la siguiente instrucción r[7]=r[9]-r[1]) y se inserta en la posición 3. El programa que resulta se muestra de la seguidamente

1. **r[9]=r[4]+r[5]**
2. **r[2]=r[6]+r[2]**
3. **r[7]=r[9]-r[1]**
4. **r[8]=sen[7]**
5. **r[1]=r[2]+r[7]**
6. **r[0]=r[1]*r[3]**

El algoritmo permite la inserción o eliminación de una instrucción como se puede comprobar. Sin embargo el intercambio de instrucciones o cambio de posición de instrucciones no hacen parte de las macro-mutaciones, el motivo se le atribuye a que estas variantes implicarían una inclusión y un borrado al mismo tiempo.

3.7.2.2 MICRO MUTACIÓN

La micro-mutación se ejecuta directamente dentro de una instrucción, y existen tres tipos de variaciones:

- Mutación de operaciones.
- Mutación de registros.
- Mutación de constantes.

Todos los registros que no se encuentren protegidos contra escritura forman el conjunto de registro destino.

Algoritmo de Micro-Mutación:

- Un programa en LGP: GP.
- Probabilidad de mutación de registros $p_{regmut.}$
- Probabilidad de mutación de la operación $p_{opermut}$
- Probabilidad de mutación de constante $p_{consmut}$
- Rango de variación de las constantes r_{const}

1. Seleccionar al azar una instrucción del programa GP.
2. Aleatoriamente seleccionar el tipo de micro-mutación (registros | operaciones | constantes) de acuerdo a las probabilidades establecidas p_{regmut} | $p_{opermut}$ | $p_{consmut}$; con $p_{regmut} + p_{opermut} + p_{consmut} = 1$.
3. Si la mutación es sobre registros:
    - Seleccionar un registro de forma aleatoria r.
    - Si r € R_{dest} (conjunto de registros destinos).
        - Seleccionar un registro destino diferente del conjunto.
    - Sino.
        - Seleccionar un registro distinto del conjunto de registro R.
4. Si la mutación es sobre operaciones:
    - Seleccionar una operación del conjunto de operaciones y reemplazar.
5. Si la mutación es sobre constantes.
    - Cambiar la constante dentro de un rango establecido r_{const}

Para la demostración se utiliza el mismo programa de entrada de la macro-mutación:

1. **r[9]=r[4]+r[5]**
2. **r[2]=r[6]+r[2]**
3. **r[8]=sen[7]**
4. **r[1]=r[2]+r[7]**
5. **r[0]=r[1]*r[3]**

Los registros existentes son 10 [0 - 9], definiendo los registros 7, 8 y 9 como las constantes. El conjunto de registros destino está compuesto por los registros [0 - 6]. Para ejecutar el punto 1 del algoritmo, se elige la instrucción 4. En el paso 2 se selecciona el tipo

de macro-mutación (en este caso registro). Para el punto 3, se selecciona el registro r[1] de la instrucción 4. Al ser un registro destino se selecciona un registro distinto, r[3]. Y en consecuencia el programa que resulta es como sigue:

1. **r[9]=r[4]+r[5]**
2. **r[2]=r[6]+r[2]**
3. **r[8]=sen[7]**
4. **r[3]=r[2]+r[7]**
5. **r[0]=r[1]*r[3]**

En la mutación de constantes puede realizarse una variante cambiando la constante por un registro. En todos los tipos de micro-mutaciones se debe asegurar el mantenimiento de la sintaxis correcta de la instrucción.

3.8 ALGORITMO UTILIZADO

Algoritmo 3.1: *Algoritmo Programación Genética Lineal*

1: Inicializar aleatoriamente una población de programas.

2: Aleatoriamente seleccionar los programas de la población y compararlos de acuerdo a su *fitness*. La medida del *fitness* define la calidad del programa en el problema que el algoritmo espera resolver.

3: Modificar los programas seleccionados utilizando alguno de los siguientes operadores de variación:

- Reproducción. Copia un programa sin cambiarlo.
- Cruzamiento. Cambia subestructuras entre dos programas.
- Mutación. Cambia un registro, un operador o una instrucción en un programa en una posición aleatoria.

4: Se construye una nueva población con los programas variados y se calcula el valor de *fitness* de los nuevos programas.

5: Si el criterio de fin no se cumple, volver a 2.

6: Parar. El mejor programa representa la mejor solución encontrada.

El *Algoritmo 3.1* fue diseñado para que logre satisfacer las necesidades del problema, que es la de lograr una predicción de ingresos de causas penales. Para ello se ha cumplido con los pasos básicos de la LGP, definiendo cada uno de ellos de acuerdo a lo referido más arriba.

Se ha realizado una adaptación de un algoritmo genérico de un Algoritmo Genético para poder implementar este trabajo.

Los individuos son una solución posible, y en el algoritmo son representados como un conjunto de instrucciones que se ejecutan en forma secuencial, a los efectos de proporcionar una posible solución al problema.

Ejemplo de individuo:

r[3] = r[2] + r[18]
r[2] = sen [1]
r[1] =r[2] + r[3]
r[0]= r[1] / r[2]

La población es el conjunto de estos individuos y en el proyecto se utiliza el sistema de generaciones, donde se construyen las nuevas poblaciones en base a la anterior, seleccionando a los padres para aplicarles los operadores evolutivos y generando con ellos los nuevos individuos.

El SelectorRuleta, es un método de selección de los individuos [8], que utiliza el mismo *fitness* asignado a cada uno de ellos como su valor de aptitud en base al peso que tienen sobre la solución del problema. Al tener el individuo un mayor valor de *fitness*, tiene mayor posibilidad de ser seleccionado al ejecutar esta forma de selección.

El mecanismo trabaja del siguiente forma, se selecciona a los padres utilizando el selector ruleta, que permite tomar en forma aleatoria a los individuos de la población, a los cuales se le aplica los operadores evolutivos de reproducción, mutación y cruzamiento si así correspondiere, y cuando se completa la cantidad establecida para la población, los padres se descartan y sigue la siguiente generación con la nueva población formada por los hijos.

Como se denota el algoritmo tiene sus límites y criterios de parada a fin de que pueda trabajar en forma. Se limita inicializando las variables al principio del algoritmo, estableciendo la cantidad de individuos “m” y la de generaciones “n”, donde el criterio de parada del algoritmo es el número máximo de generaciones establecidas.

La población inicial se crea al principio del algoritmo y se inicializa con individuos generados en forma aleatoria utilizando como base los parámetros definidos en la *Tabla 3-3*.

Esta población como se pudo observar, en cada ejecución inicial del programa genera individuos con *fitness* muy bajo para la solución del problema, y generación tras generación va mejorando con su evolución.

Con el criterio de parada establecido y la población inicial generada, se le asigna el *fitness* a cada individuo. Posteriormente se hace la selección a quienes se les aplicarán los operadores evolutivos de reproducción, cruce y mutación si así correspondiere. Generación tras generación a través del valor de aptitud se verifica cada individuo y se selecciona al mejor de cada generación para así obtener al mejor individuo como resultado del problema planteado.

En el operador evolutivo de reproducción, el individuo seleccionado pasa automáticamente a hacer parte de la siguiente población, manteniéndose intacto en el sentido de modificaciones a su estructura, esto ocurre en el caso que el individuo tenga un *fitness* que sea mejor al que actualmente se encuentre registrado como mejor individuo.

Con el cruce, se seleccionan dos individuos de la población, donde a cada individuo se le divide en sectores, tal cual se indica en la *Figura 3-1*, y se realiza el cruce intercambiando los sectores seleccionados para intercambiar, generando así los nuevos individuos para la siguiente generación.

En la mutación, ocurre de forma diferente, ya que están establecidos en el algoritmo ciertos métodos de mutación que se generan en forma aleatoria. Se establece un porcentaje de posibles mutaciones, aparte existe un criterio de cuánto por ciento de ese individuo puede ser mutado, estableciendo una cantidad fija. Asimismo para cada mutación que se realiza al individuo, el mismo puede cambiar dependiendo de cuál sea el caso, una operación, un registro o todo un cromosoma.

3.9 DISCUSIONES Y COMENTARIOS

El individuo es una solución posible, y en el caso de la LGP está representado como una serie de instrucciones imperativas orientadas a un lenguaje de programación de computadora, y que simula el lenguaje de máquina en sí. Estos programas se generan aleatoriamente en el algoritmo aplicado, donde en cada iteración evolucionan generando resultados que son comparados para lograr encontrar el mejor programa.

La finalidad de tantas poblaciones generadas es obtener al mejor individuo o en este caso el mejor resultado. La comparación se realiza por medio de un valor de aptitud o *fitness* establecido en la *ecuación 3-1*, lo cual determina si el individuo es más apto que los demás.

El individuo debe ser generado correctamente, en consecuencia se debe establecer previamente los registros (variables y constantes) y el conjunto de funciones que se podrán utilizar para generarlos. De esta estructura bien seleccionada depende la creación de programas aptos para solucionar el problema.

La cantidad máxima de iteraciones es un tema importante a definir, ya que la misma pasará a ser el criterio de parada principal del algoritmo. Por consiguiente existen otros tipos de parámetros que se constituyen como datos iniciales del programa, que contribuyen a mejorar su funcionamiento. Estos parámetros fueron puliéndose en base a las ejecuciones y experimentos que se han realizado con el LGP creado.

Los operadores evolutivos como los cruces, la mutación y la reproducción, están establecidas en las teorías, y esas teorías fueron llevadas a la práctica en el desarrollo del LGP. Teniendo toda la estructura programada, tal cual lo establece el algoritmo LGP, se procedió a ejecutar el programa, donde se obtuvieron los resultados que se exponen en el siguiente capítulo.

CAPITULO IV
METODOLOGÍA DE LAS PRUEBAS EXPERIMENTALES

4.1 INTRODUCCIÓN

En este capítulo se denota paso a paso cómo se fue desarrollando la aplicación de la metodología seleccionada para llegar a la conclusión del proyecto. El objetivo general llevó a utilizar varios medios de predicción a los efectos de compararlos y hallar el más adecuado para resolver el problema planteado. El método principal y base del estudio es el de Programación Genética Lineal desarrollado para este proyecto, los demás métodos son los estadísticos basados en serie de tiempo y su suavizamiento. Los seleccionados para este estudio fueron: Regresión Lineal, Promedio Móvil, Suavizado Exponencial y Suavizado Exponencial con Tendencia.

En la primera parte se habla del método de la programación genética lineal, explicando el lenguaje seleccionado para la programación, la forma de definición de los parámetros y los registros y operaciones utilizadas.

Adentrándose a las series temporales, se explica brevemente el concepto de las mismas, sus patrones como la tendencia, los movimientos estacionales, los movimientos cíclicos y los movimientos irregulares o aleatorios que podrían ser estudiados con este tipo de métodos. Con los métodos de series de tiempos y suavizados asignados para la implementación, se estudió y se aplicó cada una de sus fórmulas a los datos.

Las técnicas se aplicaron sobre la estructura montada y se estableció la distribución de los datos de la serie de tiempo para su estudio. Dos tipos de métricas fueron seleccionadas para evaluar el desempeño de las metodologías. Estas métricas son el error cuadrático medio para la primera métrica y el error absoluto medio para la segunda métrica.

Las tablas y gráficos comparativos se observan en este capítulo, todo en base a la implementación de cada uno de los métodos resultantes.

4.2 METODO DE PROGRAMACIÓN GENÉTICA LINEAL

El algoritmo programado es el Algoritmo 3.1, que fue descripto en el capítulo III del presente proyecto. El LGP fue desarrollado en el lenguaje de programación JAVA. Este lenguaje fue seleccionado para desarrollarlo por tener una estructura y una sintaxis similar a C y C++, aunque con un modelo de objetos más simple y eliminando las herramientas de bajo nivel. El lenguaje es orientado a objetos y tiene la posibilidad de ejecutar un mismo programa en diversas plataformas y se lo conoce por la facilidad de uso.

La LGP es un algoritmo evolutivo, en consecuencia se deben definir ciertos parámetros para que éste logre desarrollar el trabajo que se le asigna y así llegar a la solución del problema. Estos parámetros son por ejemplo: la longitud máxima de cromosomas de los programas, el tamaño de la población, las probabilidades de cruzamiento y mutación, cantidad de operadores, etc. En este trabajo se establecieron estos parámetros, y se detallaron en la *Tabla 3-3*.

La cantidad de registros quedó establecida experimentalmente en 22 registros (*Ver Tabla 3-1)*, con la siguiente nomenclatura "r[0] al r[21]", de los cuales 12 registros son constantes y 10 registros variables. Los registros r[10] al r[15] son de entrada, conteniendo información introducida desde la fuente de datos conocida, que es la serie de tiempo estudiada.

Las operaciones seleccionadas para utilizar en el proyecto fueron las aritméticas, exponenciales y las trigonométricas, totalizando la cantidad de 9 operaciones en total para la selección aleatoria por parte de cada individuo a ser ejecutado en el sistema (*Ver Tabla 3-2)* de operaciones.

4.3 SERIES TEMPORALES

Una serie temporal es un conjunto de observaciones ordenadas en el tiempo, que pueden representar la evolución de una variable (económica, física, etc.) a lo largo de él [4]. El objetivo del análisis de una serie temporal es el conocimiento de su patrón de comportamiento, para así prever su evolución futura, suponiendo que las condiciones no variarán.

Dado que no se trata de fenómenos deterministas, sino sujetos a una aleatoriedad, el estudio del comportamiento pasado ayuda a inferir la estructura que permita predecir su comportamiento futuro, pero es necesaria una gran cautela en la previsión debido a la

inestabilidad del modelo. La particular forma de la información disponible de una serie cronológica (se dispone de datos en periodos regulares de tiempo) hace que las técnicas habituales de inferencia estadística no sean válidas para estos casos, ya que nos encontramos ante "N" muestras de tamaño "i" procedentes de otras tantas poblaciones de características y distribución desconocidas.

Matemáticamente, una serie de tiempo se define por los valores Y1, Y2, Y3,…….de una variable Y (ventas mensuales, producción total, etc.) en tiempos t1, t3, t3…….. Si se reemplaza a X por la variable tiempo, estas series se definen como distribuciones de pares ordenados (X,Y) en el plano cartesiano, siendo Y una función de X; esto se denota por:

$$\mathbf{Y = f\ t\ \rightarrow Y = f(X)} \qquad (4\text{-}1)$$

El principal objetivo de las series de tiempo es hacer proyecciones o pronósticos sobre una actividad futura, suponiendo estables las condiciones y variaciones registradas hasta la fecha, lo cual permite planear y tomar decisiones a corto o largo plazo [4]. Después, con base en esa situación ideal, que supone que los factores que influyeron en la serie en el pasado lo continuarán haciendo en el futuro, se analizan las tendencias pasadas y el comportamiento de las actividades bajo la influencia de ellas (*Ver Figura 4-1*).

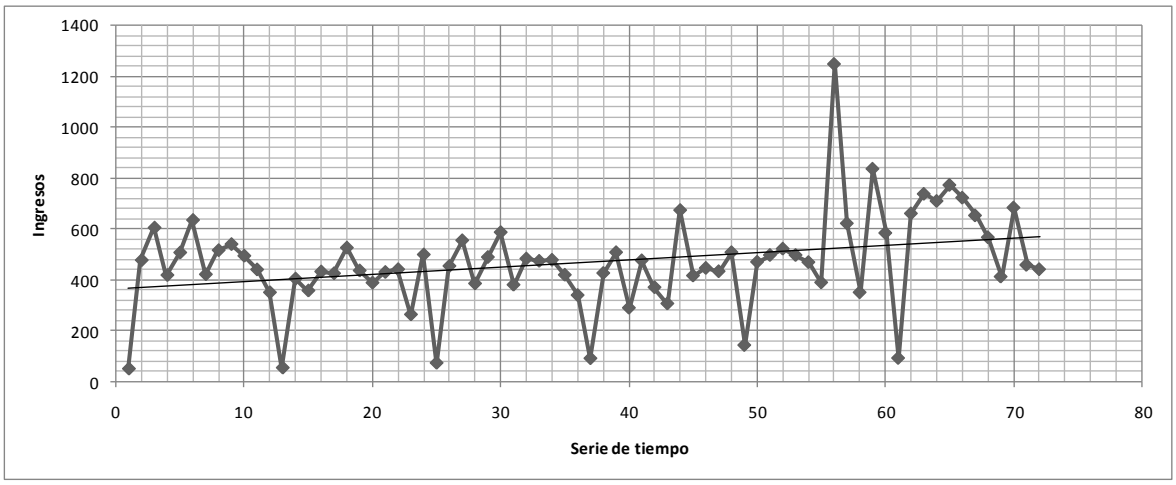


Figura 4-1 - Serie de tiempo de ingresos de causas penales [Fuente propia]

4.4 PATRONES DE SERIES DE TIEMPO

El modelo clásico o de descomposición, considera que los datos de series de tiempo están compuestas de los siguientes cuatro patrones básicos [4]:

4.4.1 TENDENCIA

La tendencia son movimientos o variaciones continuas de la variable de modo uniforme y suave, por encima o por debajo, que se observan en el largo plazo durante un período de longitud prolongada. Representan el comportamiento predominante o dirección general de la serie de tiempo como ascendente o descendente [47].

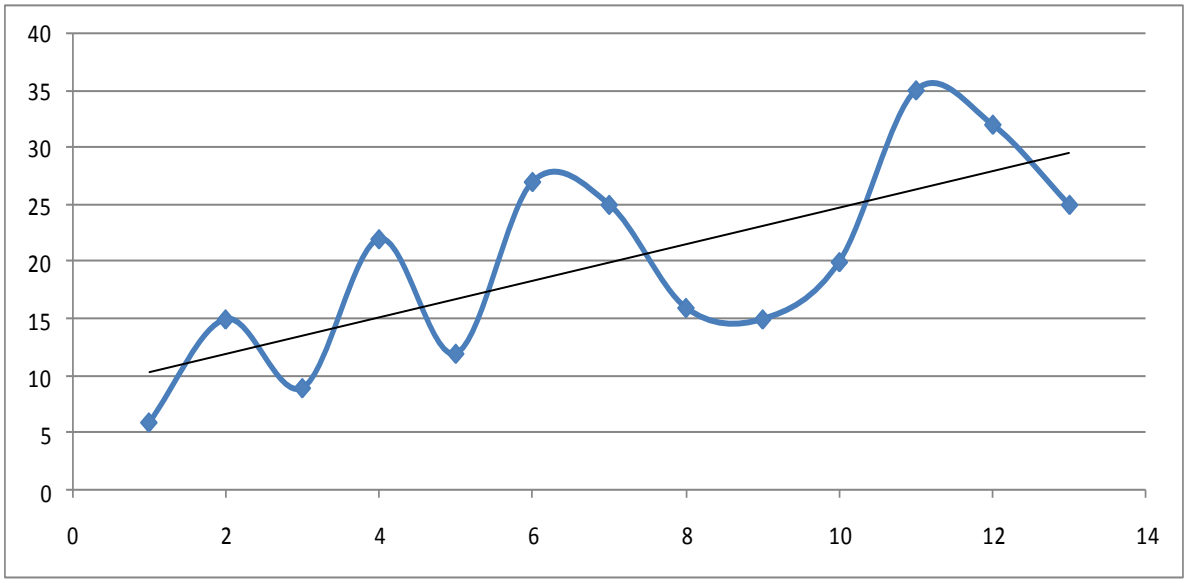


Figura 4-2 - Demanda con tendencia [Fuente propia]

4.4.2 MOVIMIENTOS ESTACIONALES

Son movimientos periódicos que se producen en forma similar cada año por la misma época, en correlación con los meses o con las estaciones del año y aun con determinadas fechas. Si los sucesos no se repiten anualmente, los datos deben recolectarse trimestral, mensual o incluso semanalmente

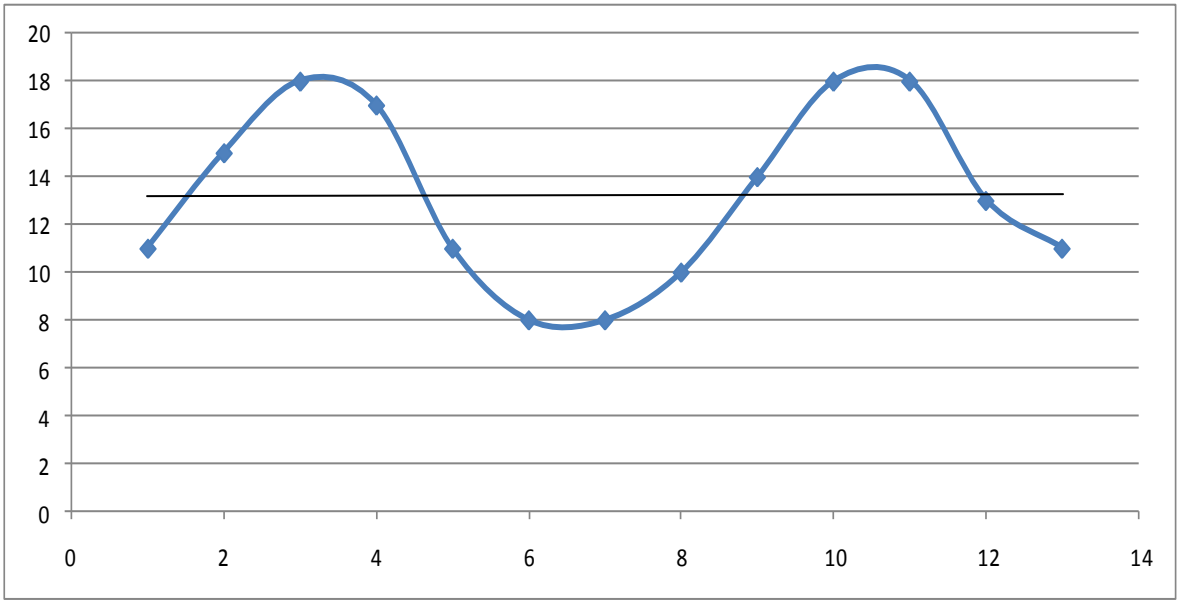


Figura 4-3 - Demanda con estacionalidad [Fuente propia]

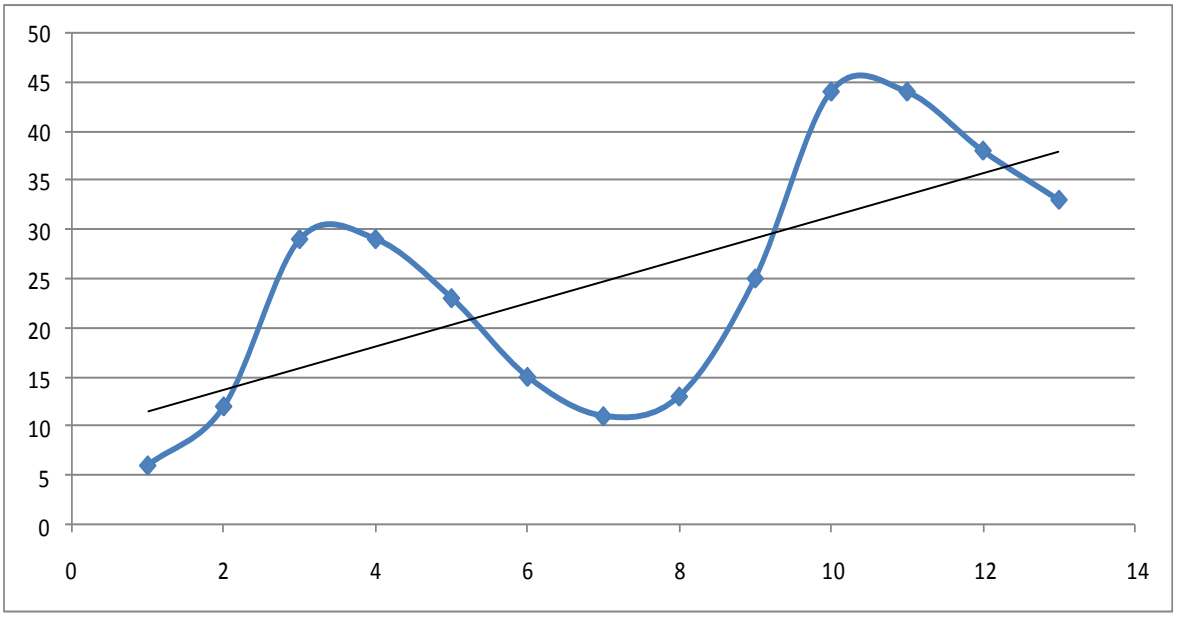


Figura 4-4 - Demanda con estacionalidad y tendencia [Fuente propia]

4.4.3 MOVIMIENTO CICLICOS

Son variaciones hacia arriba y hacia abajo de la tendencia que se presentan cada cierto número de intervalos, en forma periódica de manera ondular a modo de oscilaciones más o menos regulares durante un período relativamente prolongado, que por lo general abarca tres o más años de duración.

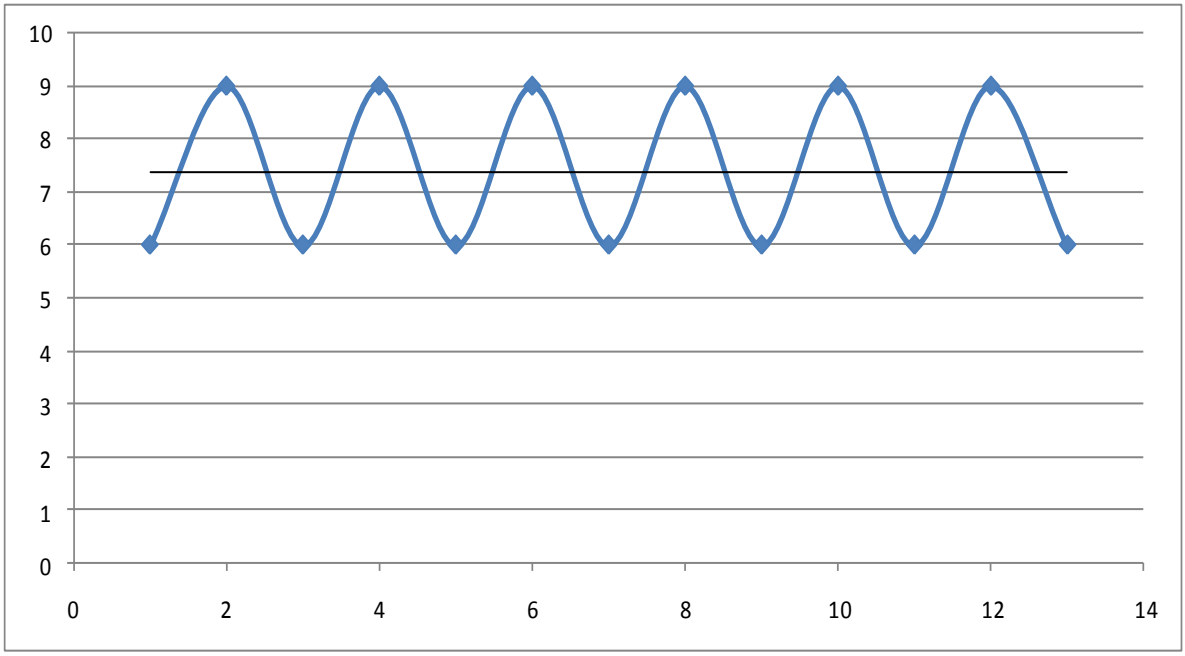


Figura 4-5 - Demanda estacionaria [Fuente propia]

4.4.4 MOVIMIENTOS IRREGULARES O ALEATORIOS

Son variaciones producidas por sucesos de ocurrencia imprevisible o accidental que producen movimientos sin un patrón discernible; así por ejemplo, las exportaciones de una empresa pueden ser afectadas por sucesos inusuales no previsibles.

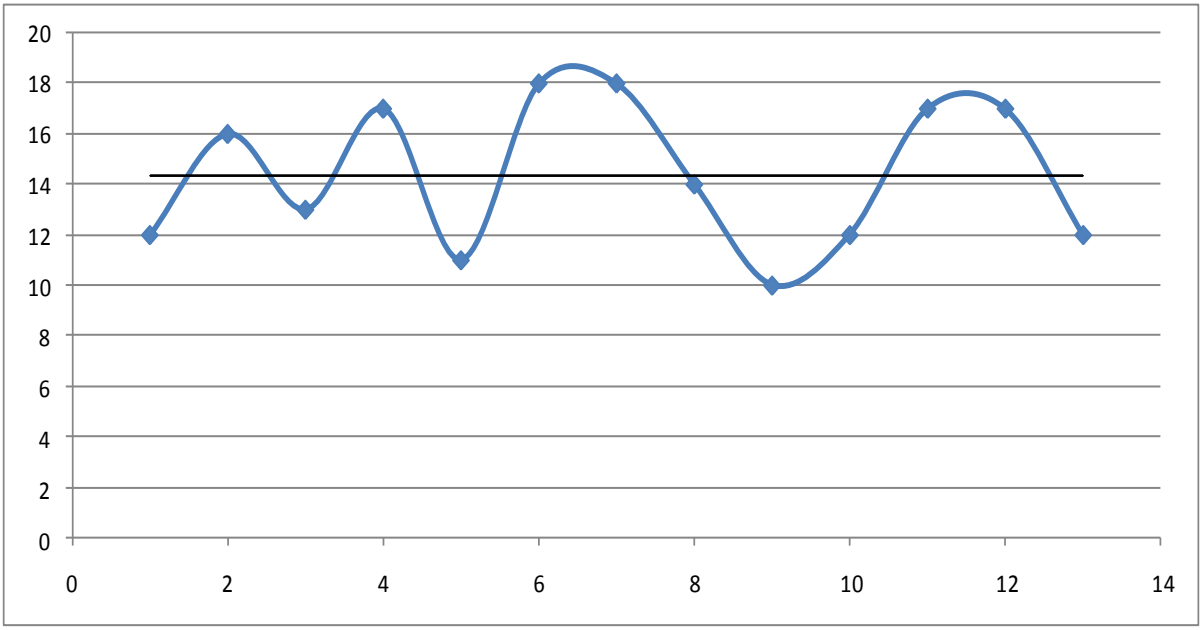


Figura 4-6 - Demanda estacionaria [Fuente propia]

4.5 SERIES DE TIEMPO SELECCIONADOS PARA EL ESTUDIO

En la planificación se ha establecido que los datos del estudio también deberían de ser aplicados a metodologías estadísticas y en este caso a métodos de serie de tiempo y suavizado, contra los que competiría el LGP, para el caso se aplicarán las fórmulas de las series de tiempo que se detallan a continuación.

4.5.1 REGRESIÓN LINEAL

En la regresión lineal simple, existe una relación entre la variable X (independiente) y la variable Y (dependiente) que puede representarse mediante una línea recta.

$$\mathbf{y = mx + b} \qquad (4-2)$$

Los valores de "b" y "m" se eligen de manera que se minimice la suma de las distancias cuadráticas entre la línea de regresión y los puntos de datos.

$$m = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{n \sum x_i^2 - (\sum x_i)^2} \qquad (4-3)$$

$$\mathbf{b = y - mx} \qquad (4-4)$$

Una buena forma de determinar si la regresión es adecuada, es a través del coeficiente de correlación, el cual mide el grado de dependencia entre los dos conjuntos de datos.

4.5.2 PROMEDIO MOVIL

Es el movimiento medio de orden "N" de una serie de valores o es el promedio aritmético de las "N" observaciones más recientes tal como se observa en su fórmula.

$$\mathbf{S_t = \frac{1}{N} \sum_{i=t}^{t-N+1} D_i} \qquad (4-5)$$

$$\mathbf{F_{t+1} = S_t} \qquad (4-6)$$

Utilizando adecuadamente los movimientos medios se eliminan las variaciones estacionales, cíclicas e irregulares, quedando sólo el movimiento de tendencia. El inconveniente de este modelo es la pérdida de datos iniciales de la serie original. Otra situación que se observa también, es que a medida que crece "N", la cantidad de datos nuevos se reduce.

4.5.3 SUAVIZADO EXPONENCIAL

Es uno de los métodos más utilizados entre las series de tiempos estacionarias. Contiene un mecanismo de corrección que ajusta los pronósticos en dirección opuesta a los

errores pasados, donde las ponderaciones disminuyen exponencialmente. Se emplea con la fórmula que sigue.

$$\mathbf{F_t} = \propto \mathbf{D_{t-1}} + (\mathbf{1} - \propto)\mathbf{F_{t-1}} \qquad (4-7)$$

α: es la constante de suavizamiento que determina la ponderación relativa puesta en la observación de demanda actual.

1-α: es el peso asignado a las observaciones pasadas de la demanda.

Algunos aspectos importantes que se deben tener en cuenta cuando se aplica el método son los siguientes:

- Aplica un conjunto de ponderaciones decrecientes a todos los datos pasados.
- La constante de suavizamiento juega el mismo papel que el "N" en los promedios móviles ("N" pequeño "α" grande asigna mayor peso a los datos actuales).
- Si "α" es grande, se da mayor ponderación a la observación actual, lo que da como resultado pronósticos que reaccionan rápidamente a los patrones de demanda, pero presentan mayor variación de periodo a periodo. Si "α" pequeño, los pronósticos son más estables.
- El mejor valor de "α" será aquel que minimice los errores.

4.5.4 SUAVIZADO EXPONENCIAL CON TENDENCIA

Este método requiere de la especificación de dos constantes de suavizamiento:

α: que suaviza el valor de la serie (promedio – estacionario)

β: que suaviza la tendencia (pendiente de los datos)

Las constantes de suavizamiento pueden ser las mismas; sin embargo, en la mayoría de las aplicaciones se da mayor estabilidad al estimado de la pendiente que al de la constante, es decir:

$$\boldsymbol{\beta} \leq \propto \qquad (4-8)$$

Por otra parte, deben utilizarse dos parámetros para la estimación del pronóstico, estos son:

S_o: corte de la recta de regresión (b)

G_o: pendiente de la recta de regresión (m)

Las ecuaciones aplicadas a este método son:

$$\mathbf{S_t = \propto D_{t-1} + (1-\propto)(S_{t-1} + T_{t-1})} \quad (4\text{-}9)$$

$$\mathbf{G_t = \beta\ S_t - S_{t-1}\ + (1-\beta)T_{t-1}} \quad (4\text{-}10)$$

$$\mathbf{F_t = S_t + T_t} \quad (4\text{-}11)$$

4.6 APLICACIÓN DE LA METODOLOGIA DESARROLLADA

Las pruebas experimentales fueron realizadas con los modelos expuestos anteriormente. Al principio se contaba con 72 meses de datos históricos de los ingresos de causas penales de los siete juzgados penales de garantías de Ciudad del Este, del periodo de tiempo de enero del 2007 a diciembre del 2012 extraídos del sitio web de la Excma. Corte Suprema de Justicia [44] y [45]. Sin embargo, con el correr del tiempo de la investigación, se ha llegado a obtener de la misma fuente de información, datos del año 2013, los cuales fueron agregados a la presente investigación a los efectos de expandir la cantidad de datos para mejorar el despliegue de las metodologías aplicadas.

Los resultados experimentales del LGP se han tratado de la siguiente manera: se utilizaron para la fase de entrenamiento 72 datos históricos. Para el periodo de validación 12 datos históricos, de los 84 datos totales con los que se contaba. Con esto se comprueba que no se ha utilizado ningún dato de entrenamiento para la correspondiente validación del LGP.

4.6.1 DISTRIBUCIÓN DE DATOS

Entonces, las pruebas del LGP se realizaron con la distribución de los datos establecidos de la siguiente forma:

- Entrenamiento o prueba : 72 meses.
- Validación o Pronóstico : 12 meses.
- Total de datos disponibles : 84 meses (enero/2007 a diciembre/2013)

4.6.2 EQUIPO UTILIZADO

Las pruebas fueron realizadas en un equipo con las siguientes características:

- Procesador : Core i3 con velocidad de 2.4 GHz.
- Memoria : 4 GB.
- Disco Duro : 500 GB.

4.6.3 MÉTRICAS DE DESEMPEÑO

Las evaluaciones de las metodologías implementadas se realizaron por métricas de desempeño. Se evaluaron en base a los errores generados por cada una de ellas, con los resultados obtenidos durante el periodo de validación en el caso del LGP y pronóstico en el caso de los métodos estadísticos.

Las técnicas seleccionadas para su abordaje son: la del error cuadrático medio y la del error absoluto medio, con las que se midieron los errores obtenidos en cada método.

4.6.3.1 PRIMERA MÉTRICA

La fórmula del error cuadrático medio aplicado a los planteamientos es la siguiente: el error es inversamente proporcional a la suma del cuadrado de la diferencia entre el valor real y pronóstico de cada periodo, considerando las "N" muestras del periodo de estudio.

$$\mathbf{e} = \frac{\mathbf{1}}{\mathbf{N}} \sum_{\mathbf{i=1}}^{\mathbf{N}} (\mathbf{x_i} - \mathbf{p_i})^{\mathbf{2}} \qquad (4\text{-}12)$$

4.6.3.2 SEGUNDA MÉTRICA

La fórmula del error absoluto medio es: el error es inversamente proporcional a la suma de la diferencia absoluta entre el valor real y el pronóstico de cada periodo, considerando las "N" muestras del periodo en estudio.

$$\mathbf{e} = \frac{\mathbf{1}}{\mathbf{N}} \sum_{\mathbf{i=1}}^{\mathbf{N}} |\mathbf{x_i} - \mathbf{p_i}| \qquad (4\text{-}13)$$

4.7 PROCEDIMIENTO DE EJECUCIÓN PARA CADA MÉTODO

Para todos los métodos LGP y Series de Tiempo:

A. Se utilizaron los 84 meses históricos de la serie de tiempo en estudio, divididos en dos partes, una de 72 meses para el entrenamiento o prueba y otra de 12 meses para la validación o pronóstico.

B. Los registros mencionados, se aplicaron a cada uno de los métodos para obtener los resultados. En el caso del LGP, la aplicación se realizó con el sistema desarrollado y en el caso de los métodos de series de tiempo, la aplicación se realizó con cada una de sus fórmulas estudiadas anteriormente en este capítulo.

C. Los resultados obtenidos en los 12 pronósticos realizados por cada método, se le aplicaron los cálculos de las métricas de los errores seleccionados, a los efectos de

obtener la información necesaria para la medición y evaluación que se realizó en las comparaciones pertinentes.

D. Por último, con los valores obtenidos de las métricas de cada método ejecutado, se procedió a la evaluación de cada uno de ellos, para obtener las conclusiones del proyecto, lo que se presenta a continuación.

Procedimiento exclusivo para el método LGP

A. La población es de estado generacional, donde los padres son reemplazados por los hijos en la siguiente generación.

B. En cada generación el sistema realiza las selecciones de los programas a través de la ruleta que tiene como método de selección y medida el *fitness* de cada uno de ellos, cuanto mayor *fitness* tienen más posibilidad de ser seleccionados. Los seleccionados serán los indicados para que se les apliquen los operadores evolutivos y así generar a los nuevos individuos de la siguiente población.

C. En cada generación se evalúa al mejor individuo entre los generados (Ver *ecuación (3-1) Fitness*), para que al finalizar la ejecución del sistema, éste pueda tener como uno de los resultados al mejor individuo de la población considerando sólo los datos de entrenamiento, que será la solución al problema planteado.

D. El pronóstico de los 12 meses correspondientes al año 2013 serán generados mediante la ejecución de la fórmula matemática obtenida con el mejor individuo seleccionado por el LGP.

4.8 ESTUDIO DE LOS RESULTADOS OBTENIDOS

Los estudios de los resultados fueron realizados por medio de tablas y gráficos que dan un mejor panorama para observar estas soluciones.

El análisis se realizó en base a las métricas establecidas donde cada tipo de error genera un resultado diferente y se puede observar en cada solución tomada como fuente de información final.

Cada metodología se estudia en forma individual, a fin de explicar la funcionalidad de cada método aplicado y por último se realiza la comparación entre todos los métodos.

4.8.1 RESULTADOS DEL LGP

La aleatoriedad del LGP para obtener soluciones distintas a cada corrida, ha generado resultados buenos en la mayoría de las ejecuciones que fueron realizadas. Los puntos a destacar son las precisiones que fueron obtenidas y para su demostración se apartaron varias generaciones y que se irán mostrando en esta parte del apartado.

Las generaciones ejecutadas, se fueron probando con variaciones de datos de entrada, registros, variables, tamaño de programas, a los efectos de encontrar la solución óptima para resolver el problema (*Ver Tabla 3-1).* En cada prueba además de encontrar resultados buenos se dio cuenta que estas soluciones simulan y logran representar a las funciones de los métodos estadísticos estudiados en este trabajo.

4.8.2 MEJORES SOLUCIONES

Los individuos obtenidos como solución en cada ejecución del programa tenían características diferentes. Los resultados siempre fueron buenos, pero los que se detallan en este trabajo fueron los mejores en base lo que se irá demostrando.

En la *Tabla 4-1*, se encuentran los códigos efectivos de cada individuo seleccionado que comprueban el excelente desempeño del LGP en la mayoría de sus corridas.

Individuo A	Individuo B	Individuo C
(Resta, 9, [15, 21]) (Suma, 0, [7, 0]) (Division, 1, [15, 19]) (Suma, 7, [1, 1]) (Abs, 0, [9]) (Suma, 8, [21, 0]) (Suma, 8, [20, 0]) (Abs, 0, [9]) (Suma, 0, [8, 1]) (Suma, 0, [7, 0]) (Suma, 8, [21, 0]) (Suma, 0, [7, 0]) (Suma, 8, [21, 0]) (Suma, 0, [7, 0]) (Suma, 8, [21, 0]) (Suma, 0, [8, 1]) (Suma, 7, [1, 1]) (Suma, 0, [7, 0]) (Suma, 8, [21, 0]) (Multiplicacion, 0, [19, 8])	(Abs, 6, [17]) (Potencia, 6, [18, 3]) (Seno, 3, [12]) (Multiplicacion, 6, [16, 11]) (Resta, 6, [17, 21]) (Resta, 2, [5, 7]) (Coseno, 6, [9]) (Abs, 0, [9]) (Seno, 3, [18]) (Ln, 3, [14]) (Ln, 4, [6]) (Seno, 9, [13]) (Ln, 9, [16]) (Multiplicacion, 5, [10, 7]) (Abs, 5, [16]) (Coseno, 0, [11]) (Coseno, 5, [19]) (Abs, 4, [15]) (Abs, 4, [19]) (Multiplicacion, 8, [14, 20]) (Seno, 2, [21]) (Division, 7, [3, 5]) (Seno, 4, [14]) (Resta, 4, [21, 15]) (Division, 2, [7, 4]) (Resta, 8, [18, 5]) (Ln, 7, [3]) (Division, 3, [9, 16]) (Resta, 0, [2, 5]) (Seno, 3, [0]) (Suma, 8, [16, 3]) (Ln, 0, [7]) (Abs, 7, [12]) (Suma, 7, [8, 9]) (Multiplicacion, 0, [7, 20])	(Suma, 4, [14, 6]) (Multiplicacion, 9, [6, 4]) (Suma, 5, [7, 17]) (Suma, 3, [13, 10]) (Seno, 3, [15]) (Suma, 5, [7, 17]) (Abs, 4, [10]) (Potencia, 0, [13, 5]) (Division, 8, [8, 7]) (Multiplicacion, 0, [7, 2]) (Resta, 0, [1, 0]) (Coseno, 4, [5]) (Resta, 6, [13, 1]) (Suma, 7, [16, 16]) (Coseno, 3, [16]) (Multiplicacion, 4, [4, 6]) (Multiplicacion, 0, [7, 2]) (Suma, 4, [14, 6]) (Suma, 5, [18, 16]) (Suma, 7, [16, 16]) (Suma, 0, [19, 12])

Tabla 4-1 - Mejores individuos de las ejecuciones del LGP [Fuente propia]

La explicación de la lectura de los individuos, se encuentra en el *Anexo 2*.

4.8.3 RESULTADO CON EL INDIVIDUO A

El mejor resultado del LGP se obtuvo en la ejecución con el Individuo A (*Ver tabla 4-2*), el error generado fue bastante bajo. El detalle de esta solución es que la misma resulta ser una línea de tendencia trazada con bastante perfección. La línea cruza generalmente un punto

medio de la demanda, lo cual hace que siempre se mantenga con un error bajo entre todos los puntos del pronóstico, lo que hace que la solución sea muy buena.

Año	Mes	X	Y Ingresos	P(A)	e^2	|e|
2013	Enero	73	244	363,84	14361,47	119,84
	Febrero	74	319	380,36	3764,75	61,36
	Marzo	75	523	383,12	19565,15	139,88
	Abril	76	489	385,89	10631,37	103,11
	Mayo	77	357	388,66	1002,25	31,66
	Junio	78	328	391,43	4022,77	63,43
	Julio	79	210	394,19	33926,80	184,19
	Agosto	80	265	396,96	17413,24	131,96
	Setiembre	81	581	399,73	32860,20	181,27
	Octubre	82	381	402,49	461,95	21,49
	Noviembre	83	471	405,26	4321,74	65,74
	Diciembre	84	362	408,03	2118,48	46,03
				SUMA TOTAL:	144450,19	1149,95
				ERROR MEDIO EN VALIDACIÓN	**12037,52**	**95,83**

Tabla 4-2 - Resultados de la ejecución 38 del LGP [Fuente propia]

4.8.3.1 PRIMERA MÉTRICA LGP INDIVIDUO A

En los cálculos del error cuadrático medio se observa que el error mantiene el promedio más bajo en comparación a los demás métodos. El menor error obtenido llega a los 461,95 puntos, y el máximo error alcanza el techo de 33926,80 puntos.

4.8.3.2 SEGUNDA MÉTRICA LGP INDIVIDUO A

Los errores obtenidos con la métrica del error absoluto medio, generó el menor error en el mes de Octubre con 21,49 puntos, y el mayor error obtenido fue en el mes de Julio con 184,19 puntos, así como en la primera métrica el promedio de errores es muy bajo.

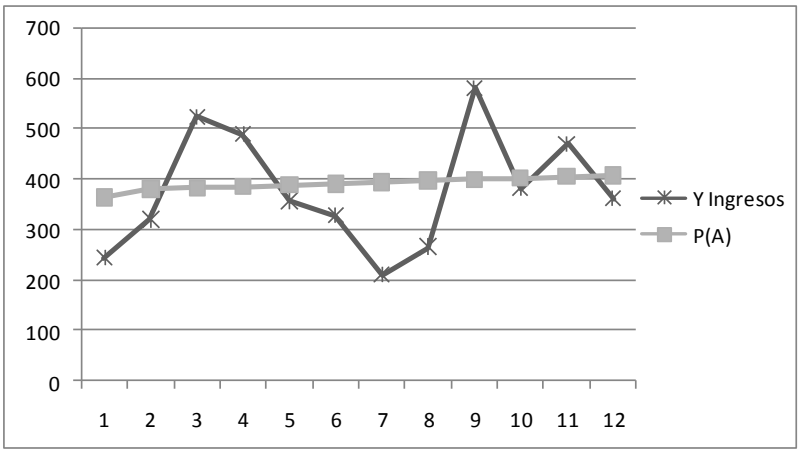


Figura 4-7 - Resultado de la ejecución 38 del LGP [Fuente propia]

Analizando la *Tabla 4-2* y la *Figura 4-7*, el LGP con el Individuo A dio este resultado lineal como mejor resultado en base a las métricas establecidas. Esto se comprueba comparando los resultados con los demás métodos. La simulación que realiza la LGP en este caso es similar a una regresión lineal, el detalle es que el LGP ha generado una fórmula capaz de conseguir los mínimos errores en cada punto de validación realizada.

4.8.4 RESULTADO CON EL INDIVIDUO B

Otro buen resultado del LGP se obtuvo con la generación del Individuo B. El individuo ejecutando en la fase de validación nos da como resultado los errores que se observan en la *Tabla 4-3* y que se pasan a estudiar a continuación.

Año	Mes	X	Y Ingresos	P(B)	e^2	|e|
2013	Enero	73	244	432,76	35628,71	188,76
	Febrero	74	319	401,52	6809,11	82,52
	Marzo	75	523	427,64	9093,21	95,36
	Abril	76	489	468,78	408,67	20,22
	Mayo	77	357	444,98	7740,42	87,98
	Junio	78	328	450,51	15009,00	122,51
	Julio	79	210	465,37	65216,27	255,37
	Agosto	80	265	450,91	34560,86	185,91
	Setiembre	81	581	457,13	15342,73	123,87
	Octubre	82	381	461,08	6413,29	80,08
	Noviembre	83	471	462,52	71,97	8,48
	Diciembre	84	362	453,63	8396,44	91,63
				SUMA TOTAL:	204690,68	1342,68
				ERROR MEDIO EN VALIDACIÓN	**17057,56**	**111,89**

Tabla 4-3 - Resultados de las métricas LGP ejecución 39 [Fuente propia]

4.8.4.1 PRIMERA MÉTRICA LGP INDIVIDUO B

En los cálculos del error cuadrático medio se observa que el error mantiene un promedio bajo en sus mejores y peores resultados, donde el máximo error en uno de sus puntos alcanza el techo de 65.216,27 puntos, siendo este un resultado bajo viendo los demás errores generados en los otros métodos. El menor resultado obtenido llega a los 71,97 puntos, siendo este uno de los errores más pequeños obtenidos con esta métrica.

4.8.4.2 SEGUNDA MÉTRICA LGP INDIVIDUO B

El error absoluto medio aplicado a los resultados del LGP, nos permite observar el acercamiento a un punto casi óptimo para un resultado en base a pronósticos, ya que los errores son mínimos, acercándose a promedios muy bajos de error.

El menor error obtenido fue en el mes de Noviembre, siendo de 8,48 puntos la diferencia entre el pronóstico obtenido como resultado y el dato de la demanda de ese mes en cuestión. En términos generales este fue uno de los menores errores obtenidos entre los resultados.

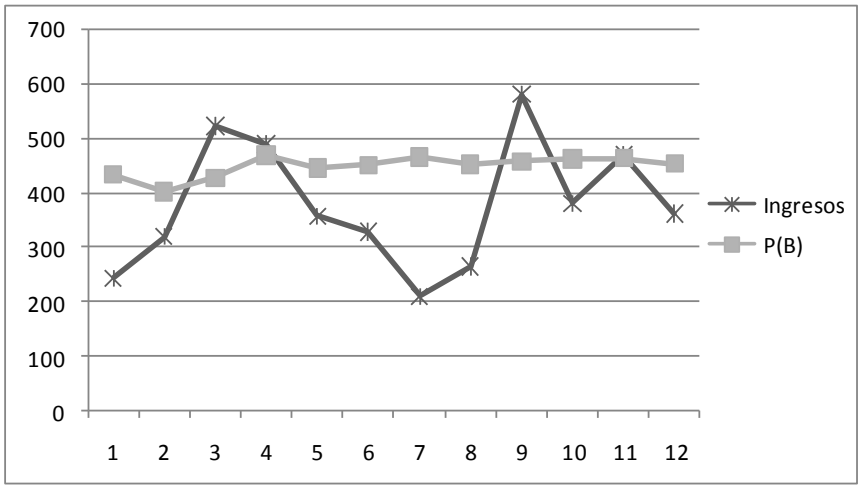


Figura 4-8 - Resultado gráfico ejecución 39 LGP [Fuente propia]

En la *Figura 4-8* de la LGP se observa una ligera tendencia de movimiento de la línea de las predicciones, la misma se mueve dentro de los puntos medios de la demanda, lo cual produce los acercamientos más significativos en varios sectores de la demanda, y en otros casos manteniendo una distancia promedio, lo cual hace que este método genere errores muy bajos, manteniendo un buen resultado en todos los puntos de sus predicciones.

4.8.5 RESULTADO CON EL INDIVIDUO C

Año	Mes	X	Y Ingresos	P(C)	e^2	|e|
2013	Enero	73	244	462,07	47552,66	218,07
	Febrero	74	319	445,07	15892,57	126,07
	Marzo	75	523	247,07	76139,73	275,93
	Abril	76	489	322,07	27867,05	166,93
	Mayo	77	357	526,07	28583,22	169,07
	Junio	78	328	492,07	26917,56	164,07
	Julio	79	210	360,07	22519,72	150,07
	Agosto	80	265	331,07	4364,68	66,07
	Setiembre	81	581	213,07	135375,64	367,93
	Octubre	82	381	268,07	12754,15	112,93
	Noviembre	83	471	584,07	12783,86	113,07
	Diciembre	84	362	384,07	486,90	22,07
				SUMA TOTAL:	411237,72	1952,26
				ERROR MEDIO EN VALIDACIÓN	**34269,81**	**162,69**

Tabla 4-4 - Resultados de las métricas LGP ejecución 39 [Fuente propia]

En la ejecución con el Individuo C (*Ver tabla 4-4*), se logró una solución que ajusta muy bien el recorrido de la serie de tiempo. Esto se dio a consecuencia de que el programa generaba predicciones cercanas de un mes, pero ajustándolo dos meses después del mes correspondiente.

4.8.5.1 PRIMERA MÉTRICA LGP INDIVIDUO C

En los cálculos del error cuadrático medio se observa el menor error obtenido por el LGP en esta métrica se genera en el mes de Diciembre con 486,90 puntos, y el máximo error alcanza en el mes de Setiembre con un techo de 135375,64 puntos que es un valor extremadamente alto. En esta ejecución el individuo generado no ha arrojado un buen resultado en base a las métricas establecidas (*Ver Tabla 4-4*).

4.8.5.2 SEGUNDA MÉTRICA LGP INDIVIDUO C

En el cálculo de la segunda métrica del error absoluto medio el menor error generado se dio en el mes de Diciembre con 22,07 puntos, y el mayor error obtenido en el mes de Setiembre con error de 367,93 puntos.

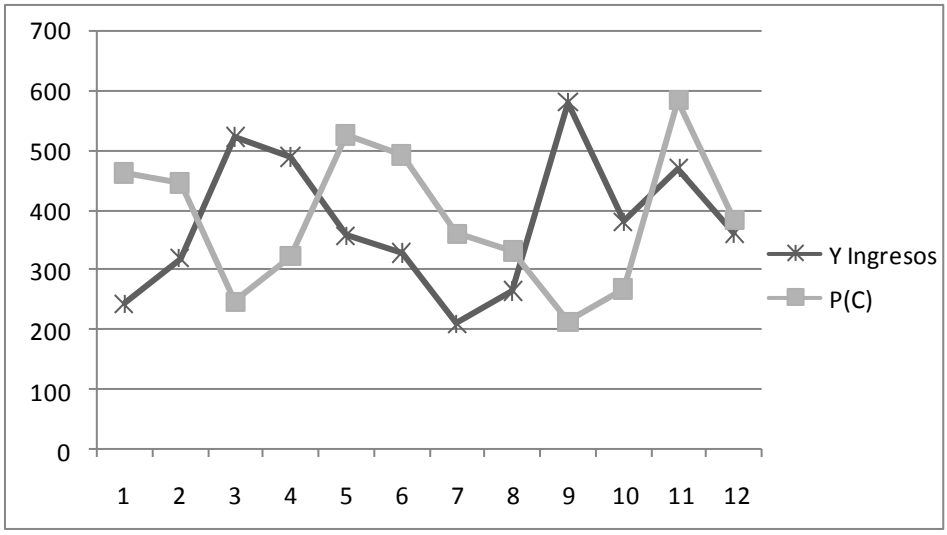


Figura 4-9 - Predicción ejecución 24 del LGP [Fuente propia]

Viendo la *Figura 4-9*, se detona que el LGP es capaz de generar resultados no lineales, y que puede acompañar a la serie de tiempo con un buen trazado en base a las predicciones obtenidas.

4.8.6 RESULTADO DE LA REGRESION LINEAL

La regresión lineal utiliza la fórmula de la recta para generar la predicción, y su función es trazar una recta lineal sin movimiento alguno entre los puntos de la demanda, esta fórmula aplicada genera resultados poco favorables para un pronóstico acertado o acercado a la demanda que se quiere pronosticar.

Este método aplicado a cualquier serie de tiempo, tendrá como resultado una recta de tendencia entre los puntos de la demanda original, y la solución a generarse sería siempre aceptable, pero sería difícil llegar a la perfección de una predicción con el mismo (*Ver Tabla 4-5*).

Año	Mes	X	Y Ingresos	P(Regresión)	e^2	|e|
2013	Enero	73	244	487,75	59413,70	243,75
	Febrero	74	319	488,79	28828,59	169,79
	Marzo	75	523	489,83	1100,22	33,17
	Abril	76	489	490,87	3,50	1,87
	Mayo	77	357	491,91	18201,12	134,91
	Junio	78	328	492,95	27209,20	164,95
	Julio	79	210	493,99	80651,84	283,99
	Agosto	80	265	495,03	52915,29	230,03
	Setiembre	81	581	496,07	7212,46	84,93
	Octubre	82	381	497,11	13482,55	116,11
	Noviembre	83	471	498,15	737,39	27,15
	Diciembre	84	362	499,20	18822,61	137,20
				SUMA TOTAL:	308578,47	1627,86
				ERROR MEDIO EN PRONÓSTICO:	**25714,87**	**135,66**

Tabla 4-5 - Resultado de las métricas Regresión Lineal [Fuente propia]

4.8.6.1 PRIMERA MÉTRICA REGRESIÓN LINEAL

El error más bajo y por ende el mejor resultado entre todas las series de tiempo fue obtenido con la regresión lineal, en base al error cuadrático medio llegó a 3,50 puntos de error, que le llevó a adjudicarse con el resultado más cercano a la demanda de ese mes. Entre la línea de sus peores pronósticos, el error más alto obtenido es el de 80.651,84 puntos. Los demás pronósticos realizados tienen un promedio aceptable. Así también se destaca que cuando la línea se aleja de los puntos de la demanda generan errores muy altos, haciendo que esta situación influya en el resultado general del método.

4.8.6.2 SEGUNDA MÉTRICA REGRESIÓN LINEAL

Coincidiendo con la métrica anterior, el menor error obtenido entre las series se adjudica también a este método, el cual se acerca a cero, que sería el resultado perfecto a obtener en este tipo de estudios. El menor error generado es de 1,87 puntos en el mes de abril de la serie estudiada.

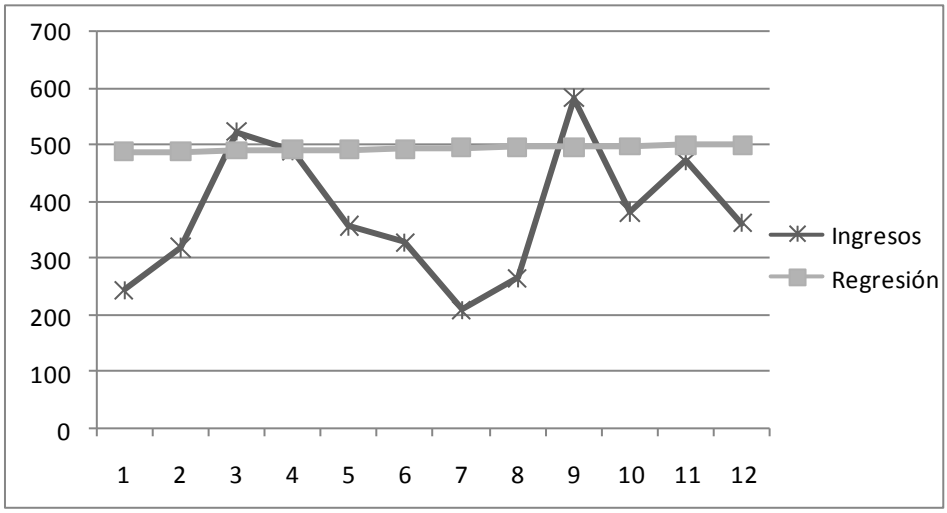


Figura 4-10 - Resultado Regresión Lineal [Fuente propia]

Como se observa en la *Figura 4-10*, la línea de regresión es una línea recta continua que nos indica la tendencia de una ligera inclinación en el crecimiento de la serie estudiada. Fuera de esa situación la recta no genera ningún movimiento aleatorio, lo que hace que este método no demuestra una efectividad a la hora de indicar un pronóstico muy acertado para poder tomar sus resultados como óptimos.

4.8.7 RESULTADO DEL PROMEDIO MÓVIL

En el método del promedio móvil, se utilizó el promedio de tres demandas anteriores para generar la predicción futura. En este caso como la técnica lo deduce, lo que se ha logrado es que los efectos estacionales o irregulares sean eliminados.

El pronóstico obtenido como resultado en base a la serie aplicada, fue aceptable, mejorando lo realizado por el método anterior. Con el promedio móvil de 3 meses se logró una aproximación en mayor medida a la demanda de nuestros datos históricos, simulando un movimiento con menos aleatoriedad a la real.

Año	Mes	X	Y Ingresos	P(P. Móvil)	e²	|e|
2013	Enero	73	244	528,67	81035,11	284,67
	Febrero	74	319	381,67	3927,11	62,67
	Marzo	75	523	335,00	35344,00	188,00
	Abril	76	489	362,00	16129,00	127,00
	Mayo	77	357	443,67	7511,11	86,67
	Junio	78	328	456,33	16469,44	128,33
	Julio	79	210	391,33	32881,78	181,33
	Agosto	80	265	298,33	1111,11	33,33
	Setiembre	81	581	267,67	98177,78	313,33
	Octubre	82	381	352,00	841,00	29,00
	Noviembre	83	471	409,00	3844,00	62,00
	Diciembre	84	362	477,67	13378,78	115,67
				SUMA TOTAL:	310650,22	1612,00
				ERROR MEDIO EN PRONÓSTICO:	**25887,52**	**134,33**

Tabla 4-6 - Resultado de las métricas Promedio Móvil [Fuente propia]

4.8.7.1 PRIMERA MÉTRICA PROMEDIO MÓVIL

Según la *Tabla 4-6*, el resultado más alto obtenido en esta métrica es de 98.177,78 puntos, lo que demuestra que genera errores muy altos para una predicción. Generando errores altos hace que el resultado en forma general no sea bueno. El error más bajo obtenido alcanza los 841,00 puntos, lo cual demuestra que se encuentra lejos de los mejores resultados obtenidos con otros métodos.

4.8.7.2 SEGUNDA MÉTRICA PROMEDIO MÓVIL

El menor valor obtenido fue de 29,00 puntos y el mayor valor generado es de 313,33 puntos, se observa la misma situación que la anterior métrica, donde ambos puntos se encuentran lejos de ser los mejores resultados obtenidos entre los demás métodos.

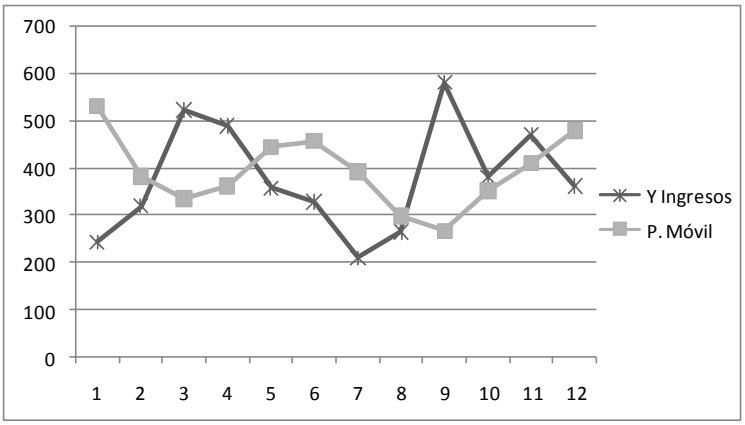


Figura 4-11 - Resultado Promedio Móvil [Fuente propia]

En la *Figura 4-11* se observa cómo se generan los cortes de los picos altos de la demanda, el cual crea movimientos uniformes al pronóstico.

4.8.8 SUAVIZADO EXPONENCIAL

El objetivo principal de esta metodología, es la corrección y ajuste de los pronósticos, para lograr que los resultados obtenidos sigan en dirección opuesta a los errores pasados.

En este caso se ha logrado pronosticar obteniendo los mejores resultados, utilizando el método de suavización exponencial aplicando un alpha igual a:

$$\alpha = 0{,}4$$

Con estos resultados se encontró dificultad para competir, ya que los errores generados por este método se encuentran entre los mejores. Sus promedios obtenidos son los más bajos entre los métodos estadísticos.

Año	Mes	X	Y Ingresos	P(S. Expon.)	e^2	|e|
2013	Enero	73	244	504,13	67667,37	260,13
	Febrero	74	319	400,08	6573,60	81,08
	Marzo	75	523	367,65	24134,67	155,35
	Abril	76	489	429,79	3506,06	59,21
	Mayo	77	357	453,47	9307,00	96,47
	Junio	78	328	414,88	7548,77	86,88
	Julio	79	210	380,13	28944,29	170,13
	Agosto	80	265	312,08	2216,35	47,08
	Setiembre	81	581	293,25	82801,86	287,75
	Octubre	82	381	408,35	747,92	27,35
	Noviembre	83	471	397,41	5415,65	73,59
	Diciembre	84	362	426,85	4204,92	64,85
				SUMA TOTAL:	243068,46	1409,88
				ERROR MEDIO EN PRONÓSTICO:	**20255,70**	**117,49**

Tabla 4-7 - Resultado de las métricas Suavizado Exponencial [Fuente propia]

4.8.8.1 PRIMERA MÉTRICA SUAVIZAMIENTO EXPONENCIAL

Dentro de la métrica del error cuadrático medio, lo que se puede observar de los resultados obtenidos por este método (*Tabla 4-7*) es que por más que genere excelentes resultados a nivel general, lo que pasa es que cuando tiende a una falla, genera errores muy altos por lo que su mayor error obtenido es de 82.801,86 puntos.

Asimismo se observa que el menor error obtenido fue de 747,92 puntos, lo que le deja también lejos de ser mejor error entre el menor obtenido entre los demás métodos. Sin embargo en la mayoría de sus predicciones este método genera errores promedio muy bajos, lo que hace que sea muy buena su utilización para realizar predicciones.

4.8.8.2 SEGUNDA MÉTRICA SUAVIZAMIENTO EXPONENCIAL

El menor error generado fue de 27,35 puntos y el mayor 287,75, como ocurre en la métrica anterior los errores no son los mejores resultados, pero si genera una tendencia de promedio muy baja, lo cual hace que este sea uno de los mejores métodos de pronósticos que se puede utilizar con fin de evaluar soluciones futuras.

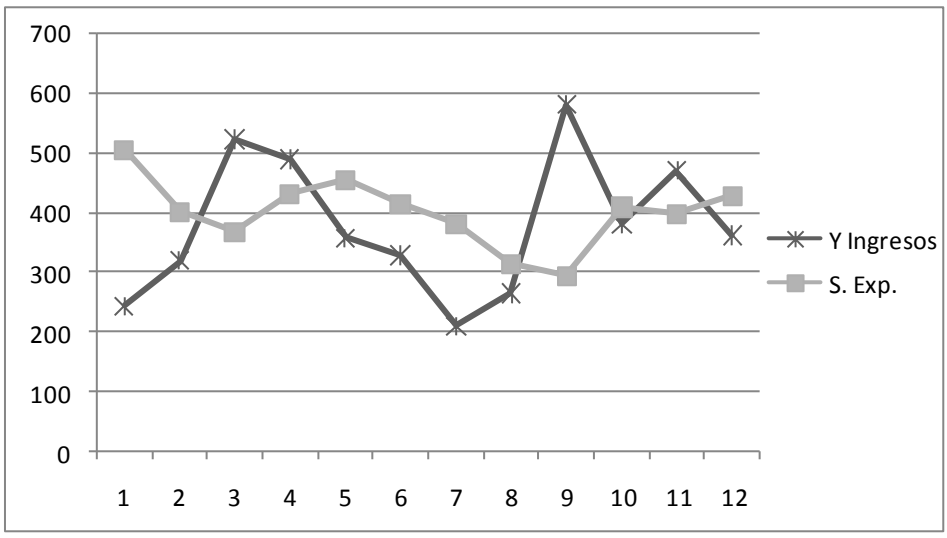


Figura 4-12 - Resultado suavizado exponencial [Fuente propia]

Los movimientos obtenidos en la *Figura 4-12* son muy buenos, tratan de simular los movimientos de la serie de tiempo estudiada, y como la técnica es de suavizar, incluye los suavizamientos sobre los picos altos de la serie.

4.8.9 SUAVIZADO EXPONENCIAL CON TENDENCIA

En este método se ha realizado primeramente el cálculo de la suavización exponencial simple, que se aplica en el método anterior, y luego se aplica con otro cálculo la suavización exponencial sobre los datos resultantes de la primera, para ello se utilizaron las fórmulas establecidas en el *punto 4.5.4*.

En este caso para la primera fórmula se ha aplicado un valor alpha y para la segunda fórmula un valor beta con los siguientes valores:

$$\alpha = 0,5$$

$$\beta = 0,1$$

Estos valores son las suavizamientos utilizados y con los que mejores resultados han obtenidos para esta serie de tiempo.

Año	Mes	X	Y Ingresos	P(S. Exp. Tend.)	e^2	|e|
2013	Enero	73	244	473,39	52617,80	229,39
	Febrero	74	319	336,78	316,19	17,78
	Marzo	75	523	305,09	47484,53	217,91
	Abril	76	489	402,14	7544,58	86,86
	Mayo	77	357	438,01	6562,36	81,01
	Junio	78	328	385,89	3351,48	57,89
	Julio	79	210	342,44	17540,12	132,44
	Agosto	80	265	255,09	98,19	9,91
	Setiembre	81	581	239,41	116682,34	341,59
	Octubre	82	381	406,65	658,03	25,65
	Noviembre	83	471	388,99	6725,72	82,01
	Diciembre	84	362	429,26	4523,74	67,26
				SUMA TOTAL:	264105,08	1349,69
				ERROR MEDIO EN PRONÓSTICO:	**22008,76**	**112,47**

Tabla 4-8 - Resultados de las métricas suavizado exponencial con tendencia [Fuente propia]

4.8.9.1 PRIMERA MÉTRICA SUAVIZACION EXPONENCIAL CON TENDENCIA

Los resultados (*Tabla 4-8*) obtenidos por este método se encuentran entre los primeros por haber obtenido una solución general óptima, siendo una metodología de predicción muy buena por cómo logra suavizar y obtener la tendencia de la serie de tiempo sobre la que se aplica.

Sin embargo dentro de esta métrica, es el que ha generado el error más alto en uno de los puntos de pronóstico. En el mes de setiembre alcanzó un error de 116.682,34 puntos, un error muy alto para lograr satisfacer un pronóstico exacto, sin embargo en los demás puntos si obtuvo acercamientos razonables, por lo que ha logrado a nivel general un buen resultado.

4.8.9.2 SEGUNDA MÉTRICA SUAVIZACIÓN EXPONENCIAL CON TENDENCIA

Para la métrica del error absoluto medio, el menor error obtenido fue de 9,91 puntos y el mayor error de 341,59 puntos. Esta técnica ha generado en los demás puntos errores promedios muy bajos.

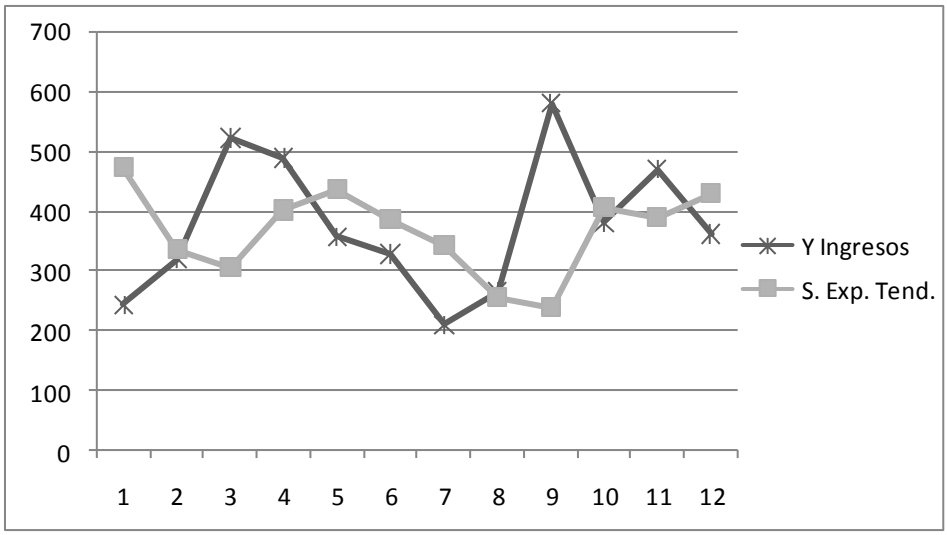


Figura 4-13 - Resultado suavizado exponencial con tendencia [Fuente propia]

La *Figura 4-13* del suavizado exponencial con tendencia, genera un trayecto muy bueno simulando el recorrido de nuestra serie de tiempo original, no es perfecto, pero si es uno de los que más se ha ajustado en ese punto de vista.

4.9 COMPARACIÓN Y EVALUACIÓN DE RESULTADOS

Los métodos de LGP y los métodos estadísticos aplicados en este proyecto han arrojado resultados muy interesantes. El objetivo principal del proyecto fue la de ejecutar cada una de estas metodologías y comparar las soluciones que obtuvieron en base a la serie de tiempo de los ingresos de causas penales de los Juzgados Penales de Garantías de Ciudad del Este y la comparación se realiza a continuación.

4.9.1 PRIMERA MÉTRICA DEL ERROR CUADRÁTICO MEDIO

Para observar mejor los resultados obtenidos con la métrica del error cuadrático medio, se ha montado la *Tabla 4-9* donde las soluciones de cada una de las técnicas ejecutadas se han asentado.

Tabla de Error Cuadrático Medio							
Mes	**e² PGL(A)**	**e² PGL(B)**	**e²(S. Exp.)**	**e²(S. Exp. Tend.)**	**e²(Regresión)**	**e²(P. Móvil)**	**e² PGL(C)**
Enero	14361,47	35628,71	67667,37	52617,80	59413,70	81035,11	47552,66
Febrero	3764,75	6809,11	6573,60	316,19	28828,59	3927,11	15892,57
Marzo	19565,15	9093,21	24134,67	47484,53	1100,22	35344,00	76139,73
Abril	10631,37	408,67	3506,06	7544,58	3,50	16129,00	27867,05
Mayo	1002,25	7740,42	9307,00	6562,36	18201,12	7511,11	28583,22
Junio	4022,77	15009,00	7548,77	3351,48	27209,20	16469,44	26917,56
Julio	33926,80	65216,27	28944,29	17540,12	80651,84	32881,78	22519,72
Agosto	17413,24	34560,86	2216,35	98,19	52915,29	1111,11	4364,68
Setiembre	32860,20	15342,73	82801,86	116682,34	7212,46	98177,78	135375,64
Octubre	461,95	6413,29	747,92	658,03	13482,55	841,00	12754,15
Noviembre	4321,74	71,97	5415,65	6725,72	737,39	3844,00	12783,86
Diciembre	2118,48	8396,44	4204,92	4523,74	18822,61	13378,78	486,90
TOTAL Error:	144450,19	204690,68	243068,46	264105,08	308578,47	310650,22	411237,72
Error Medio:	12037,52	17057,56	20255,70	22008,76	25714,87	25887,52	34269,81

Tabla 4-9 - Comparación de resultados con error cuadrático medio [Fuente propia]

Analizando la *Tabla 4-9*, el resultado final del error deja en primer y segundo lugar con los mejores resultados al LGP con los Individuos A y B, con un total de 12.037,52 y 17.057,56 puntos respectivamente, quedando en tercer lugar el método de suavizado exponencial con un total de 20.255,70 puntos y en cuarto lugar el método suavizado exponencial con tendencia con un total de 22.008,76 puntos.

Los errores obtenidos con la métrica del error cuadrático medio, muestran que la LGP con los Individuos A y B obtuvieron un comportamiento constante, siempre quedando cada resultado entre los mejores lugares, o sea con errores bajos en comparación a los demás métodos.

4.9.2 SEGUNDA MÉTRICA DEL ERROR ABSOLUTO MEDIO

La métrica del error medio absoluto nos da la opción de certificar si la técnica que resulta con mejores resultados en la primera métrica, es la misma o varía con este otro modo de observar el error que tiene cada solución.

Tabla de Error Absoluto Medio							
Mes	**|e| PGL(A)**	**|e| PGL(B)**	**|e|(S. Exp. Tend.)**	**|e|(S. Exp.)**	**|e|(P. Móvil)**	**|e|(Regresión)**	**|e| PGL(C)**
Enero	119,84	188,76	229,39	260,13	284,67	243,75	218,07
Febrero	61,36	82,52	17,78	81,08	62,67	169,79	126,07
Marzo	139,88	95,36	217,91	155,35	188,00	33,17	275,93
Abril	103,11	20,22	86,86	59,21	127,00	1,87	166,93
Mayo	31,66	87,98	81,01	96,47	86,67	134,91	169,07
Junio	63,43	122,51	57,89	86,88	128,33	164,95	164,07
Julio	184,19	255,37	132,44	170,13	181,33	283,99	150,07
Agosto	131,96	185,91	9,91	47,08	33,33	230,03	66,07
Setiembre	181,27	123,87	341,59	287,75	313,33	84,93	367,93
Octubre	21,49	80,08	25,65	27,35	29,00	116,11	112,93
Noviembre	65,74	8,48	82,01	73,59	62,00	27,15	113,07
Diciembre	46,03	91,63	67,26	64,85	115,67	137,20	22,07
TOTAL Error:	1149,95	1342,68	1349,69	1409,88	1612,00	1627,86	1952,26
Error Medio:	95,83	111,89	112,47	117,49	134,33	135,66	162,69

Tabla 4-10 - Comparación de resultados con error absoluto medio [Fuente propia]

Observando la *Tabla 4-10*, se da cuenta que el método que genera los mejores resultados quedando en primer lugar es el LGP con sus Individuos A y B, con un error total de 95,83 y 111,89 puntos respectivamente. Asimismo en los siguientes lugares si hubo variación comparando con la anterior métrica, y en este caso el tercer lugar queda a favor del método de suavizado exponencial con tendencia con un error total de 112,47 puntos, y el cuarto lugar en base a los mejores resultados queda para el método de suavizado exponencial con error total de 117,49 puntos.

Con el error absoluto medio se da cuenta que los errores generados por el programa LGP con sus Individuos A y B, mantienen un promedio bajo en todos los puntos validados, lo cual es importante destacar, ya que a consecuencia de ser constante en sus errores, sus resultados son siempre cercanos a las demandas estudiadas, y esto hizo que el método haya tenido éxito en sus predicciones.

El programa LGP presentó en dos casos los mejores resultados para la serie de tiempo de los ingresos de causas penales de los Juzgados de Garantías de Ciudad del Este, por lo que se encuentra comprobada la eficiencia de la metodología con estas pruebas realizadas.

El método LGP tiene la capacidad de generar fórmulas matemáticas aleatorias. En este caso se ha comprobado que debido a su aleatoriedad, los resultados no siempre serán buenos, como se pudo observaren el Individuo C, a través de las métricas establecidas. Sin embargo, no todo se echa a perder con este individuo, ya que fue capaz de generar un buen recorrido gráfico en comparación a la serie de tiempo estudiada. Entonces se puede deducir que el LGP es capaz de obtener resultados muy buenos para series de tiempo no lineales o irregulares.

4.10 APRENDIZAJE DEL LGP

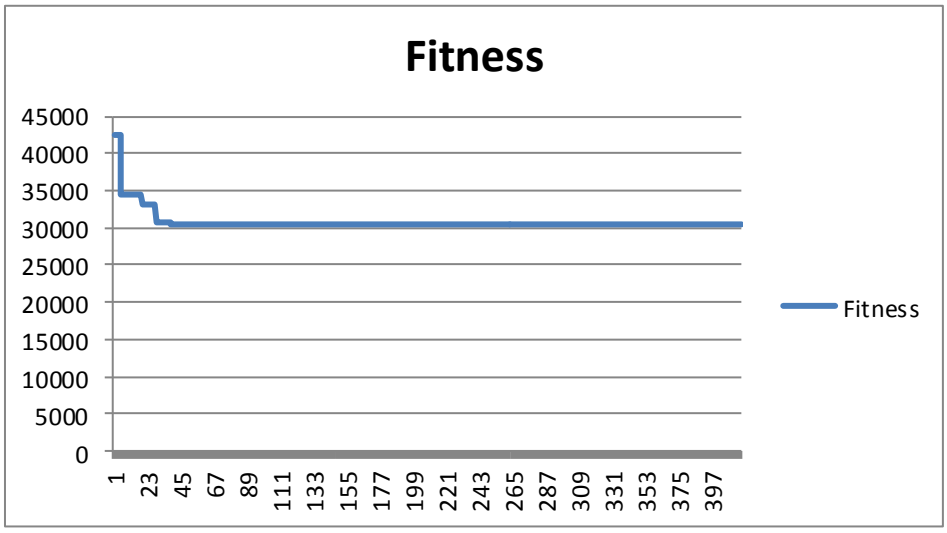


Figura 4-14 - Demostración del aprendizaje del LGP [Fuente propia]

A través de cada generación, tal como se comprueba con la *Figura 4-14*, el programa va aprendiendo en base a sus errores. El *fitness* es el gran artífice de este aprendizaje, ya que por medio de esta medida de aptitud que se le asigna a cada individuo de la población (Ver ecuación (*3-1*)), se conoce qué tan bien el mismo logra los resultados. En cada comparación que realiza durante el funcionamiento del sistema, el mejor individuo se va almacenando hasta llegar al criterio de parada del programa, obteniendo como resultado al mejor individuo.

4.11 DISCUSIONES Y COMENTARIOS

En este capítulo se fue demostrando el desarrollo realizado para aplicar cada una de las metodologías que se utilizaron para llegar al resultado final. En la primera parte se hizo un pequeño resumen de la Programación Genética Lineal y de las series de tiempo explicando los patrones de las mismas.

Las fórmulas de los métodos estadísticos: la Regresión Lineal, el Promedio Móvil, el Suavizado Exponencial y el Suavizado Exponencial con Tendencia, fueron transcriptas y explicadas para su aplicación en este capítulo.

La metodología aplicada para desarrollar este trabajo fue elaborada en base a la organización de los datos, las métricas de desempeños seleccionadas, posteriormente se asignaron los procedimientos de ejecución que se cumplieron para cada método, dejando aclarado cuales serían de uso exclusivo para el LGP y cuáles de aplicación general.

Continuando con el proceso, los resultados obtenidos se demostraron con el estudio pormenorizado de los mismos. La revisión en forma individual de cada uno de estos resultados se realizó evaluando con las métricas de desempeño.

Las tablas y gráficos se han elaborado a los efectos de representar los resultados, para dar un mejor panorama a toda la aplicación de la metodología, por último se evaluaron y se realizaron las comparaciones de todos los métodos, del cual resultó ganador el LGP.

Las reglas obtenidas por el LGP se pueden observar en la *Tabla 4-1*, estas son las fórmulas matemáticas halladas por el LGP y que fueron los artífices de los resultados obtenidos.

CAPITULO V
CONCLUSIONES Y LÍNEAS FUTURAS DE TRABAJO

5.1 RESULTADOS Y CONCLUSIONES

En distintos campos viene siendo comprobado que la predicción mediante programación genética lineal ha adquirido una relevante importancia dentro del área de estudios de series de tiempo, utilizando datos históricos con diferentes tendencias y movimientos.

Con la presente tesis se ha evaluado la predicción de ingresos de causas penales mediante programación genética lineal, analizando y comparándolo con otros métodos estadísticos a través de dos tipos de métricas. Se han realizado los ensayos prácticos sobre esa serie de tiempo a través de un algoritmo LGP desarrollado en el lenguaje JAVA, y aplicándolos también con las fórmulas de los métodos de regresión lineal, promedio móvil, suavizado exponencial y suavizado exponencial con tendencia.

En este capítulo se recopilan los resultados y conclusiones más relevantes de esta tesis, que se pueden separar en tres grupos:

5.2 CONCLUSIONES METODOLÓGICAS

Dentro de esta parte se incluyen aquellos aspectos generales que se pueden aplicar a cualquier caso que se pueda tratar a futuro y que hacen referencia al método de trabajo y el área del estudio.

- Continuando el área de investigación abierta en la tesis de Roberto Sánchez [1], se ha realizado el experimento, analizando la serie de tiempo de ingresos de causas penales de los Juzgados de Garantías de Ciudad del Este, utilizando esta metodología de la LGP, se concluye que la técnica es apta para ser aplicada en cualquier serie de tiempo porque genera excelentes resultados, como se pudo comprobar en esta línea de datos jurídicos a la que fue aplicada. La mayor dificultad que se tuvo con esta serie, fue su comportamiento aleatorio, y que resulta ser totalmente diferente a las series de tiempo estudiadas por Roberto Sánchez [1]. Con lo que se tiene por cumplido el objetivo general (A) y el objetivo específico (a).
- Los resultados de los métodos LGP, regresión lineal, promedio móvil, suavizado exponencial y suavizado exponencial con tendencia, fueron aplicados con total éxito, los cuales pudieron ser llevados a comparación sin objeción alguna. Objetivo específico (b).

- El algoritmo programado, permite que los datos de la serie de tiempo puedan ser variados o reemplazados, ya sea parcial o totalmente, a los efectos de que se puedan aplicar a series de tiempos similares y el mismo seguirá prediciendo sin problema, o sea metodológicamente cambiando los datos históricos del sistema, el mismo podrá predecir la nueva serie sin mayores dificultades, con lo que se ha logrado que el sistema pueda predecir datos similares de los ingresos de causas penales de cualquier región del país. Objetivo específico (c).

5.3 CONCLUSIONES DE TENDENCIA

En un punto del estudio uno de los objetivos fue la de determinar la tendencia de los ingresos para años posteriores a los experimentados, el mismo LGP se encargó a través de sus resultados generados responder esta situación, obteniendo una información precisa como resultado para responder a esta pregunta.

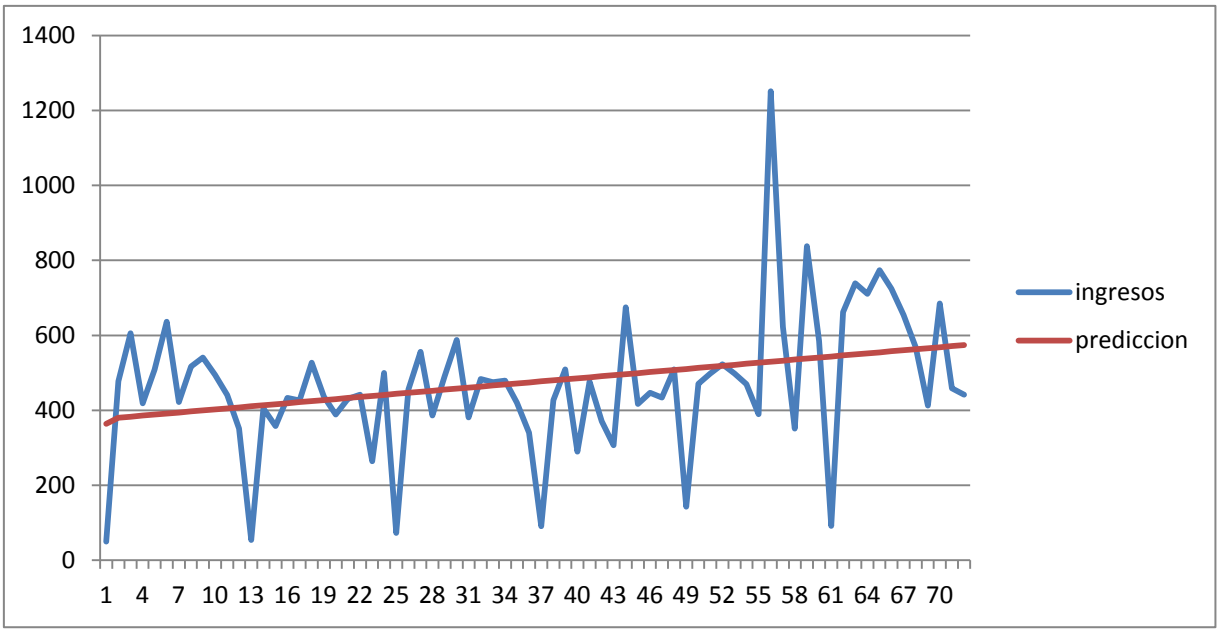


Figura 5-1 - Tendencia de crecimiento de los ingresos [Fuente propia]

La tendencia de los ingresos penales es de un ligero crecimiento como se puede observar en la *Figura 5-1*, el cual como ya se había manifestado, fue fruto de uno de los buenos resultados obtenidos como solución en una de las corridas del LGP. En este caso el programa actuó generando la tendencia de crecimiento de esta serie de tiempo. Objetivo específico (d).

Observando este crecimiento en esta área de los delitos, así como se puede ver el incremento en la estructura de la Excma. Corte Suprema de Justicia [44] y [45], esta viene

creciendo año tras año tanto en infraestructura como en recurso humano, entonces se puede concluir que con este tipo de investigaciones, la Corte puede realizar una reingeniería de sus recursos, llevando infraestructura y recurso humano donde más se necesite. En consecuencia aplicando este tipo de técnicas y tecnologías obtendría mejores resultados en el ámbito de la justicia, ya que ajustando los procesos podrá ofrecer un mejor servicio en el área en la que se encuentra especializada, LA JUSTICIA. Objetivo específico (e, f).

5.4 CONCLUSIONES Y RESULTADOS DE RELEVANCIA

A través de esta investigación se ha logrado aplicar la LGP para elaborar un modelo que permitió realizar predicciones de una variable jurídica, al cual se le ha realizado las pruebas experimentales y comparaciones con otros modelos matemáticos. Para la comparación se utilizaron dos tipos de métricas, la del error cuadrático medio y la del error absoluto medio. Objetivo específico (g)

- La LGP fue ejecutada varias veces y en la mayoría de las ejecuciones generaba resultados muy buenos. Observando estos resultados y comparándolos con cada modelo matemático, se dio cuenta que las soluciones simulan y logran representar a las soluciones obtenidas por estos métodos matemáticos, con la gran diferencia que el LGP logra generalmente mejores resultados.

- Para la primera métrica del error cuadrático medio, dos resultados del LGP obtuvieron los mejores desempeños quedando en primer y segundo lugar (*Tabla 5-1*).

Posición	Método	Error Cuadrático Medio
1	PGL Individuo A	12.037,52
2	PGL Individuo B	17.057,56
3	Suavizado exponencial	20.255,70
4	Suavizado exponencial con tendencia	22.008,76
5	Regresión Lineal	25.714,87
6	Promedio Móvil	25.887,52

Tabla 5-1 - Tabla de posicionamiento de la primera métrica [Fuente propia]

- Para la segunda métrica del error absoluto medio, nuevamente la LGP ha obtenido los mejores resultados, quedando con los Individuos A y B en el primer y segundo lugar (*Tabla 5-2*).

Posición	Método	Error Absoluto Medio
1	PGL Individuo A	95,83
2	PGL Individuo B	111,89
3	Suavizado exponencial con tendencia	112,47
4	Suavizado exponencial	117,49
5	Promedio Móvil	134,33
6	Regresión Lineal	135,66

Tabla 5-2 - Tabla de posicionamiento de la segunda métrica [Fuente propia]

El éxito del LGP radica en que los resultados que genera se mantienen con un promedio de error bajo, quedando siempre entre los primeros lugares en cada predicción realizada, y al efectuar las sumatorias de los errores y aplicar las métricas asignadas, no varía sus resultados para ninguna de ellas. Sin embargo como se vio entre los resultados matemáticos, hubo cambio de posición entre los dos mejores resultados entre cada métrica, esto obedece a que sus resultados generan en ciertos puntos de predicción errores muy altos que afectan a su desempeño general.

Las variantes que se le pueden aplicar a la LGP para su ejecución permiten obtener resultados para modelos de predicciones lineales y no lineales, otro de los motivos por lo que genera soluciones eficientes en comparación a los métodos estadísticos clásicos que demuestran limitaciones para predicciones de series de tiempos no lineales [48].

Entonces como conclusión final, la LGP es apta para ser aplicada como metodología de predicción de los ingresos de causas penales de los Juzgados de Garantías de Ciudad del Este, por la eficiencia obtenida en sus resultados en base a esta serie de tiempo estudiada en este proyecto, inclusive pudiendo ser extendida su aplicación para otras regiones del país.

5.5 TRABAJOS FUTUROS

Los trabajos futuros que se podrían continuar en vista de la investigación realizada en el presente proyecto, se proponen seguidamente:

1. Publicar los resultados obtenidos en congresos y revistas nacionales y/o internacionales.
2. Análisis de parámetros de inicialización y los valores de entrada.
3. Comparación de la LGP con otros modelos no lineales, como las redes neuronales.
4. Estudiar problemas con múltiples salidas.
5. Implementación de la LGP con enfoques multi-objetivo.

REFERENCIAS

REFERENCIAS

[1] Sánchez, R., "Aplicación De La Programación Genética Lineal Para La Predicción De Indicadores Macro Económicos", Proyecto Final De Tesis, Facultad De Ciencias Y Tecnología Universidad Católica Nuestra Señora De La Asunción. 2008.

[2] Bhattacharyya, S., Pictet, O., Zumbach, G., "*Knowledge-Intensive Genetic Discovery InForeign Exchange Markets*", IEEE Transactions On Evolutionary Computation, Vol. 6, N°. 2, 2002.

[3] Zumbach, G., Pictet. O., Y Masutti, O., "*Genetic Programming WithSyntactic Restrictions Applied To Financial Volatility Forecasting*", Olsen & Associatesresearch Institute For Applied Economics. 2001.

[4] Jenkins, M. "*Time Series Analysis Forecasting And Control*", 2da Edition, Holden-Day, San Francisco, 1976.

[5] Santini, M., Tettamanzi, A. "*Genetic Programming For Financial Time Series Prediction", "Genetic Programming, Proceedings Of Eurogp'2001*", Vol. 2038, 2001.

[6] Darwin, Ch. "*The Origin Of Species By Means Of Natural Selection Or The Preservation Of Favored Races In The Struggle For Life*". Random House, New York, 1993.

[7] Holland, J. H. "*Adaptation In Natural An Artificial Systems". MIT Press, Cambridge, Massachusetts*, 2da. Edition, 1992.

[8] Koza, J. R. "*Herarchical Genetic Algorithms Operating On Populations Of Computer Programs*". In N. S. Sridharan, Editor, Proceedings Of 11th International Joint Conference On Artificial Intelligence, San Mateo, Morgan Kaufmann, California, 1989.

[9] Cramer, Nichael Lynn (1985), "*A representation for the Adaptive Generation of Simple Sequential Programs*" in Proceedings of an International Conference on Genetic Algorithms and the Applications, Grefenstette, John J. (ed.), Carnegie Mellon University

[10] Koza, J. R. (2010). *Human-competitive results produced by genetic programming*. *Genetic Programming and Evolvable Machines*, *11*(3-4), 251-284.

[11] Koza, J. R. (2010, July). *Introduction to genetic programming tutorial: from the basics to human-competitive results. In Proceedings of the 12th annual conference companion on Genetic and evolutionary computation* (pp. 2137-2262). ACM.

[12] Poli, R., & Koza, J. (2014). *Genetic Programming* (pp. 143-185). Springer US.

[13] Langdon, W. B., McKay, R. I., & Spector, L. (2010). *Genetic programming. In Handbook of Metaheuristics* (pp. 185-225). Springer US.

[14] Mora García, A. M., Castillo Valdivieso, P. A., Merelo Guervós, J. J., Alfaro Cid, E., Esparcia-Alcázar, A. I., & Sharman, K. (2008, July). *Discovering causes of financial distress by combining evolutionary algorithms and artificial neural networks. In Proceedings of the 10th annual conference on Genetic and evolutionary computation* (pp. 1243-1250). ACM.

[15] Nasfi, R., & Soui, M. (2014). *Extraction of Interesting Adaptation Rules. Procedia Computer Science*, 34, 607-612.

[16] Martínez, C. A., & Velásquez-Henao, J. D. (2013). Una modificación de la metodología de regresión simbólica para la predicción de series de tiempo. Ingeniería y Universidad, 17(2), 325-338.

[17] Raffo Lecca, E., & Ruiz Lizama, E. (2014). Optimización por computación evolucionaria. Industrial Data, 8(2), 061-068.

[18] Dominguez Catena. Tesis de Final de Grado, T. F. Regresión Generalizada a través de Programación Genética.

[19] Puente-Montejano, C. A., & Olague-Caballero, G. Estimación De Erosión Con El Modelo Rusle Sin Sobre-Estimar El Factor De Cobertura Usando Indices De Vegetación Generados Artificialmente.

[20] Yubisleydis, R. P., & Leoandris, S. V. (2010, November). SLD257-"Plug-ins para la integración de los módulos de predicción de actividad biológica por Programación Genética y Árboles de Regresión a la plataforma Grato". In VIII Congreso Internacional de Informática en la Salud. II Congreso Moodle Salud.

[21] Ascari, R. E. D. O. S., Borsoi, B. T., & Fávero, E. M. D. B. (2013). Algoritmo evolutivo para predição de dados antropométricos infantis como apoio à tomada de decisão. Journal of Health Informatics, 5(1).

[22] Catoira, A., Pérez, J. L., Rabuñal, J. R., & Martínez-Abella, F. Regresión simbólica en Ingeniería Civil mediante técnicas de Programación Genética.

[23] Romero Morales, C., Soto, S. V., & De Castro, C. (2013). Aplicación de Algoritmos Evolutivos como Técnica de Minería de Datos para la Mejora de

Cursos Hipermedia Adaptativos basados en Web. RIED. Revista Iberoamericana de Educación a Distancia, 6(2).

[24] Escalante, H. J., Mendoza, K., Graff, M., & Morales-Reyes, A. (2013). *Genetic Programming of Prototypes for Pattern Classification. In Pattern Recognition and Image Analysis* (pp. 100-107). Springer Berlin Heidelberg.

[25] Ali Ghorbani, M., Khatibi, R., Aytek, A., Makarynskyy, O., & Shiri, J. (2010). *Sea water level forecasting using genetic programming and comparing the performance with artificial neural networks. Computers & Geosciences*, 36(5), 620-627.

[26] Nitsure, S. P., Londhe, S. N., & Khare, K. C. (2014). *Prediction of sea water levels using wind information and soft computing techniques. Applied Ocean Research*, 47, 344-351.

[27] Lee, Y. S., & Tong, L. I. (2011). *Forecasting energy consumption using a grey model improved by incorporating genetic programming. Energy conversion and Managemen*t, 52(1), 147-152.

[28] Shiri, J., & KişI, Ö. (2011). *Comparison of genetic programming with neuro-fuzzy systems for predicting short-term water table depth fluctuations. Computers & Geosciences*, 37(10), 1692-1701.

[29] Chen, S. H., & Chen, J. N. (2010). *Forecasting container throughputs at ports using genetic programming. Expert Systems with Applications*, 37(3), 2054-2058.

[30] Azamathulla, H. M., & Ghani, A. A. (2011). *Genetic programming for predicting longitudinal dispersion coefficients in streams. Water resources management*, 25(6), 1537-1544.

[31] Ngomo, A. C. N., & Lyko, K. (2012). *Eagle: Efficient active learning of link specifications using genetic programming. In The Semantic Web: Research and Applications* (pp. 149-163). Springer Berlin Heidelberg.

[32] Azamathulla, H. M., Guven, A., & Demir, Y. K. (2011). *Linear genetic programming to scour below submerged pipeline. Ocean Engineering*, 38(8), 995-1000.

[33] Guven, A., & Kişi, Ö. (2011). *Daily pan evaporation modeling using linear genetic programming technique. Irrigation science*, 29(2), 135-145.

[34] Gandomi, A. H., Alavi, A. H., & Sahab, M. G. (2010). *New formulation for compressive strength of CFRP confined concrete cylinders using linear genetic programming. Materials and Structures*, 43(7), 963-983.

[35] Wilson, G., & Banzhaf, W. (2010, July). *Interday foreign exchange trading using linear genetic programming. In Proceedings of the 12th annual conference on Genetic and evolutionary computation* (pp. 1139-1146). ACM.

[36] Kisi, O., & Guven, A. (2010). *A machine code-based genetic programming for suspended sediment concentration estimation. Advances in Engineering Software*, 41(7), 939-945.

[37] Uyumaz, A., Mehr, A. D., Kahya, E., & Erdem, H. (2014). *Rectangular side weirs discharge coefficient estimation in circular channels using linear genetic programming approach*.

[38] Wilson, G., Leblanc, D., & Banzhaf, W. (2011, July). *Stock trading using linear genetic programming with multiple time frames. In Proceedings of the 13th annual conference on Genetic and evolutionary computation* (pp. 1667-1674). ACM.

[39] Alavi, A. H., Aminian, P., Gandomi, A. H., & Esmaeili, M. A. (2011). *Genetic-based modeling of uplift capacity of suction caissons. Expert Systems with Applications*, 38(10), 12608-12618.

[40] Alavi, A. H., Gandomi, A. H., & Mollahasani, A. (2012). *A Genetic Programming-Based Approach for the Performance Characteristics Assessment of Stabilized Soil. In Variants of Evolutionary Algorithms for Real-World Applications* (pp. 343-376). Springer Berlin Heidelberg.

[41] Shavandi, H., & Ramyani, S. S. (2013). *A linear genetic programming approach for the prediction of solar global radiation. Neural Computing and Applications*, 23(3-4), 1197-1204.

[42] Alavi, A. H., & Gandomi, A. H. (2012). *Energy-based numerical models for assessment of soil liquefaction. Geoscience Frontiers*, 3(4), 541-555.

[43] Yaghouby, F., Ayatollahi, A., Bahramali, R., & Yaghouby, M. (2012). *Robust genetic programming-based detection of atrial fibrillation using RR intervals. Expert Systems*, 29(2), 183-199.

[44] Http://Www.Csj.Gov.Py/, Sitio De La Excma. Corte Suprema De Justicia

[45] Http://Www.Pj.Gov.Py/, Sitio Del Poder Judicial

[46] Brameier, M. Y Banzhaf, W., "*A Comparison Of Linear Genetic Programming And Neural Networks In Medical Data Mining*". IEEE Transactions On Evolutionary Computation, Vol. 5, 2001.

[47] Martínez, F., "Análisis de las Series Temporales de los Precios del Mercado Eléctrico mediante Técnicas de Clustering", Universidad de Sevillla - España, 2005.

[48] Dabhi, V. K., & Chaudhary, S. (2014, February). *Time Series Modeling and Prediction Using Postfix Genetic Programming. In Advanced Computing & Communication Technologies* (ACCT), 2014 Fourth International Conference on(pp. 307-314). IEEE.

[49] Koza, J. R., "*Genetic Programming: On the Programming of Computers by Means of Natural Selection*", MIT Press, Cambridge, MA, USA, 1992.

[50] Hillier, L. "Investigación de Operaciones", Séptima Edición, McGraw Hill, México, 2002.

[51] Brameier, "*On Linear Genetic Programming*", PhD Thesis, University of Dortmund, Dortmund, Alemania, Feb. 2004.

[52] Aikenhead, M., "*Legal Knowledge Based Systems: Some Observations On The Future*", First Published In Web Journal Of Current Legal Issuesin Association With Blackstone Press Ltd., 1995.

[53] Borgulya, I. Two "*Examples Of Decision Support In The Law. Artificial Intelligence And Law*", Vol. 7, 1999.

[54] Greenfield,J., "*Decision Support WithinCriminal Justice System*", Proceedings Of The 13th BILETA Conference: "*The Changing Jurisdiction*", 1998.

[55] Engle, E. A. "*An Introduction To Artificial Intelligence And Legal Reasoning: Using Xtalk To Model The Alien Tort Claims Act And Torture Victim Protection Act. St. John's Journal OfLegal Commentary*", Vol. 11, No. 2, 2004.

[56] Back, T., Hammel, U., y Schwefel, H. "*Evolutionary computation: Comments on the history and current state*". En Fogel, D.B. (Ed.) 1997.

[57] Biethanhn, J. y Nissen, V. (Co-Eds) "*Evolutionary Algorithms in Management Applications*". Springer, New York, 1995.

[58] Davis, L. "*Handbook of Genetics Algorithms*", Ban Nostrand, New York, 1991.

ANEXO 1

Planilla de los ingresos de causas penales de los siete juzgados de garantías de Ciudad del Este, periodo de fecha Enero/2007 a Diciembre/2013.

Año	Meses	X	Y Ingresos
2007	Enero	1	50
	Febrero	2	478
	Marzo	3	606
	Abril	4	419
	Mayo	5	508
	Junio	6	636
	Julio	7	422
	Agosto	8	517
	Setiembre	9	541
	Octubre	10	495
	Noviembre	11	441
	Diciembre	12	351
2008	Enero	13	54
	Febrero	14	405
	Marzo	15	358
	Abril	16	433
	Mayo	17	426
	Junio	18	527
	Julio	19	437
	Agosto	20	389
	Setiembre	21	431
	Octubre	22	442
	Noviembre	23	264
	Diciembre	24	500
2009	Enero	25	73
	Febrero	26	455
	Marzo	27	556
	Abril	28	386
	Mayo	29	490
	Junio	30	588
	Julio	31	381
	Agosto	32	484
	Setiembre	33	475
	Octubre	34	479
	Noviembre	35	420
	Diciembre	36	340
2010	Enero	37	91
	Febrero	38	427
	Marzo	39	509

	Abril	40	290
	Mayo	41	477
	Junio	42	371
	Julio	43	307
	Agosto	44	675
	Setiembre	45	417
	Octubre	46	447
	Noviembre	47	434
	Diciembre	48	509
2011	Enero	49	143
	Febrero	50	471
	Marzo	51	498
	Abril	52	523
	Mayo	53	498
	Junio	54	470
	Julio	55	390
	Agosto	56	1251
	Setiembre	57	623
	Octubre	58	351
	Noviembre	59	838
	Diciembre	60	585
2012	Enero	61	92
	Febrero	62	662
	Marzo	63	739
	Abril	64	711
	Mayo	65	774
	Junio	66	724
	Julio	67	654
	Agosto	68	568
	Setiembre	69	413
	Octubre	70	685
	Noviembre	71	459
	Diciembre	72	442
2013	Enero	73	244
	Febrero	74	319
	Marzo	75	523
	Abril	76	489
	Mayo	77	357
	Junio	78	328
	Julio	79	210
	Agosto	80	265
	Setiembre	81	581
	Octubre	82	381
	Noviembre	83	471
	Diciembre	84	362

ANEXO 2

Los individuos interpretados en formato de registros se describen a continuación.

Individuo A	Individuo B	Individuo C
r[9] = r[15] - r[21]	r[6] = |r[17]|	r[4] = r[14] + r[6]
r[0] = r [7] + r[0]	r[6] = r [18] ^ r[3]	r[9] = r[6] * r[4]
r[1] = r[15] / r[19]	r[3] = sen[12]	r[5] = r[7] + r[17]
r[7] = r[1] + r[1]	r[6] = r[16] * r[11]	r[3] = r[13] + r[10]
r[0] = |r[9]|	r[6] = r[17] - r[21]	r[3] = sen[15]
r[8] = r[21] + r[0]	r[2] = r[5] - r[7]	r[5] = r[7] + r[17]
r[8] = r[20] + r[0]	r[6] = cosen[9]	r[4] = |r[10]|
r[0] = |r[9]|	r[0] = |r[9]|	r[0] = r [13] ^ r[5]
r[0] = r [8] + r[1]	r[3] = sen[18]	r[8] = r[8] / r[7]
r[0] = r [7] + r[0]	r[3] = ln[14]	r[0] = r[7] * r[2]
r[8] = r[21] + r[0]	r[4] = ln[6]	r[0] = r[1] - r[0]
r[0] = r [7] + r[0]	r[9] = sen[13]	r[4] = cosen[5]
r[8] = r[21] + r[0]	r[9] = ln[16]	r[6] = r[13] - r[1]
r[0] = r [7] + r[0]	r[5] = r[10] * r[7]	r[7] = r [16] + r[16]
r[8] = r[21] + r[0]	r[5] = |r[16]|	r[3] = cosen[16]
r[0] = r [8] + r[1]	r[0] = cosen[11]	r[4] = r[4] * r[6]
r[7] = r[1] + r[1]	r[5] = cosen[19]	r[0] = r[7] * r[2]
r[0] = r [7] + r[0]	r[4] = |r[15]|	r[4] = r [14] + r[6]
r[8] = r[21] + r[0]	r[4] = |r[19]|	r[5] = r [18] + r[16]
r[0] = r[19] * r[8]	r[8] = r[14] * r[20]	r[7] = r [16] + r[16]
	r[2] = sen[21]	r[0] = r [19] + r[12]
	r[7] = r[3] / r[5]	
	r[4] = sen[14]	
	r[4] = r[21] - r[15]	
	r[2] = r[7] / r[4]	
	r[8] = r[18] - r[5]	
	r[7] = ln[3]	
	r[3] = r[9] / r[16]	
	r[0] = r[2] - r[5]	
	r[3] = sen[0]	
	r[8] = r[16] + r[3]	
	r[0] = ln[7]	
	r[7] = |r[12]|	
	r[7] = r[8] + r[9]	
	r[0] = r[7] * r[20]	

ANEXO 3

Los códigos fuentes de las clases principales del algoritmo programado se anexan a continuación.

CREACIÓN DEL PROGRAMA

```
package ae;

import java.util.Arrays;
import java.util.List;
import java.util.Set;

import lib_util.log.Errors;
import lib_util.util.Factory;
import lib_util.util.Utilities;
importLGP.Operacion;
import datos.Datos;

public class Programa implements Cloneable, Comparable<Programa>
{
	private List<Instruccion> cromosoma;
	private List<Instruccion> efectivos;
	private Double fitness;
	private boolean fitnessCalculado;
	private boolean efectivosCalculados;

	public Programa()
	{
		super();

		this.fitnessCalculado = false;
		this.efectivosCalculados = false;

		this.generarCromosoma();
	}

	public void avisarModificado()
	{
		this.fitnessCalculado = false;
		this.efectivosCalculados = false;
	}

	public Programa(List<Instruccion> cromosoma)
	{
		super();

		this.fitnessCalculado = false;
		this.efectivosCalculados = false;

		this.setCromosoma(cromosoma);
	}

	private void generarCromosoma()
	{
		this.fitnessCalculado = false;
		this.efectivosCalculados = false;

		int programSize = Utilities.irand(Datos.MIN_CANTIDAD_CROMOSOMAS_INICIALES,
Datos.MAX_CANTIDAD_CROMOSOMAS_INICIALES);

		List<Instruccion> cromosoma = Factory.getLinkedList();

		for(int i = 0; i < programSize; i++)
		{
			Operacion operacion = Datos.obtenerOperacion();

			int indiceDestino = Utilities.irand(0, Datos.UMBRAL_CONSTANTES - 1);
			int[] indiceOperandos = new int[operacion.getCantidadOperandos()];

			for(int j = 0; j < indiceOperandos.length; j++)
			{
				indiceOperandos[j] = Utilities.irand(0, Datos.CANTIDAD_REGISTROS
- 1);
```

```
                        }

                        if(i == programSize - 1) indiceDestino = 0;

                        cromosoma.add(new Instruccion(operacion, indiceDestino,
indiceOperandos));
                }

                this.setCromosoma(cromosoma);
        }

        public List<Instruccion> getCromosoma()
        {
                if(this.cromosoma == null) return null;

                List<Instruccion> cromosomaClonado = Factory.getLinkedList();

                for(Instruccion instruccion : this.cromosoma)
                {
                        Instruccion clonInstruccion = instruccion.copiar();

                        cromosomaClonado.add(clonInstruccion);
                }

                return cromosomaClonado;
        }

        public void setCromosoma(List<Instruccion> cromosoma)
        {
                this.cromosoma = cromosoma;

                this.fitnessCalculado = false;
                this.efectivosCalculados = false;
        }

        private void establecerEfectivos(List<Instruccion> cromosoma)
        {
                List<Instruccion> efectivos = Factory.getLinkedList();
                boolean[] efectividad = new boolean[cromosoma.size()];

                marcarEfectivos(cromosoma, efectividad);

                for(int i = 0; i < efectividad.length; i++)
                {
                        if(efectividad[i])
                        {
                                efectivos.add(cromosoma.get(i));
                        }
                }

                this.setEfectivos(efectivos);

                this.efectivosCalculados = true;
        }



        private void marcarEfectivos(List<Instruccion> cromosoma2, boolean[] efectividad)
        {
                Set<Integer> registrosRelevantes = Factory.getHashSet();

                registrosRelevantes.add(0);

                for(int i = efectividad.length - 1; i >= 0; i--)
                {
                        Instruccion instruccion = cromosoma.get(i);
                        int indiceDestino = instruccion.getIndiceDestino();

                        efectividad[i] = registrosRelevantes.contains(indiceDestino);

                        if(efectividad[i])
                        {
                                registrosRelevantes.remove(indiceDestino);

                                for(int indiceOperando : instruccion.getIndiceOperandos())
                                {
                                        registrosRelevantes.add(indiceOperando);
                                }
```

```
                }
        }
}

public void setCromosoma(Instruccion... cromosoma)
{
        this.setCromosoma(Factory.arrayToList(cromosoma));
        this.fitnessCalculado = false;
        this.efectivosCalculados = false;
}

private void calcularFintess()
{
        Fitness.calcularFitness(this);
        this.fitnessCalculado = true;
}

public Double getFitness()
{
        if(!this.fitnessCalculado)
                this.calcularFintess();

        return this.fitness;
}

void setFitness(Double fitness)
{
        this.fitness = fitness;
}

public double getPrediccion(double[] registros)
{
        return this.calcularPrediccion(registros);
}

public List<Instruccion> getEfectivos()
{
        if(efectivos == null || !this.efectivosCalculados)
                this.establecerEfectivos(this.getCromosoma());

        return efectivos;
}

private void setEfectivos(List<Instruccion> efectivos)
{
        this.efectivos = efectivos;
}

public Programa copiar()
{
        try
        {
                return this.clone();
        }
        catch(CloneNotSupportedException e)
        {
                Errors.message(e);
                return null;
        }
}

@Override
public Programa clone() throws CloneNotSupportedException
{
        Programa clon = (Programa) super.clone();

        List<Instruccion> nuevoCromosoma = Factory.getLinkedList();

        for(Instruccion instruccion : this.cromosoma)
        {
                nuevoCromosoma.add(instruccion);

        }

        clon.cromosoma = nuevoCromosoma;

        if(this.efectivos != null)
        {
```

```
                        List<Instruccion> nuevoEfectivos = Factory.getLinkedList();

                        for(Instruccion instruccion : this.efectivos)
                        {
                                nuevoEfectivos.add(instruccion);
                        }

                        clon.efectivos = nuevoEfectivos;
                }

                return clon;
        }

        private double calcularPrediccion(double[] registros)
        {
                for(Instruccion instruccion : this.getEfectivos())
                {
                        instruccion.ejecutarInstruccion(registros);
                }

                return registros[0];
        }

        @Override
        public String toString()
        {
                StringBuilder sb = new StringBuilder();

                sb.append(Arrays.toString(this.getCromosoma().toArray()));
                if(this.fitnessCalculado)
                {
                        sb.append("\n");
                        sb.append("Fitness: ").append(this.getFitness());
                }


                return sb.toString();
        }

        @Override
        public int compareTo(Programa comparado)
        {
                if(comparado == null) return 1;

                Double fitness1 = this.getFitness();
                Double fitness2 = comparado.getFitness();

                if(fitness1 == null) return (fitness2 == null) ? 0 : -1;
                if(fitness2 == null) return 1;

                if(fitness1.isNaN()) return (fitness2.isNaN()) ? 0 : -1;
                if(fitness2.isNaN()) return 1;

                if(fitness1.isInfinite() && fitness1 > 0) return (fitness2.isInfinite() &&
fitness2 > 0) ? 0 : 1;
                if(fitness2.isInfinite() && fitness2 > 0) return -1;

                if(fitness1.isInfinite() && fitness1 < 0) return (fitness2.isInfinite() &&
fitness2 < 0) ? 0 : -1;
                if(fitness2.isInfinite()  && fitness2 < 0) return 1;

                int resultado = 0;

                if(fitness1 > fitness2) resultado = 1;
                if(fitness1 < fitness2) resultado = -1;

                return resultado;
        }

        @Override
        public boolean equals(Object obj)
        {
                if(this == obj) return true;

                if(obj == null || !obj.getClass().equals(this.getClass())) return false;

                Programa programa = (Programa) obj;
```

```
		Instruccion[] cromosoma1 = this.getCromosoma().toArray(new
Instruccion[this.getCromosoma().size()]);
		Instruccion[] cromosoma2 = programa.getCromosoma().toArray(new
Instruccion[programa.getCromosoma().size()]);

		if(cromosoma1.length != cromosoma2.length) return false;

		for(int i = 0; i < cromosoma1.length; i++)
		{
			if(!Utilities.equals(cromosoma1[i], cromosoma2[i])) return false;
		}

		return true;
	}
}



package ae;

import java.util.Arrays;

import lib_util.util.DescendentComparator;
import lib_util.util.Utilities;
import datos.Datos;

public class Programas
{
	private Programa[] poblacion;
	private Programa[] poblacionElite;
	private Selector selector;

	public Programas()
	{
		this.selector = new SelectorRuleta(new MutacionOperador(), new
CruceProgramas());
		this.generarPoblacion();
	}

	public Programa[] getPoblacion()
	{
		return this.poblacion;
	}

	public void setPoblacion(Programa[] poblacion)
	{
		this.poblacion = poblacion;
	}

	public Programa[] getPoblacionElite()
	{
		return this.poblacionElite;
	}

	public void setPoblacionElite(Programa[] poblacionElite)
	{
		this.poblacionElite = poblacionElite;
	}

	public void generarPoblacion()
	{
		if(this.poblacion == null)
			this.poblacion = new Programa[Datos.TAM_POBLACION];

		for(int i = 0; i < this.poblacion.length; i++)
		{
			this.poblacion[i] = new Programa();
		}

		this.poblacionElite = this.selector.guardarElite(this.poblacionElite,
this.poblacion);
	}

	public void mutar()
	{
```

```
		this.poblacion = this.selector.mutarPoblacion(this.poblacion);
		this.poblacionElite = this.selector.guardarElite(this.poblacionElite,
this.poblacion);
	}

	public void mutacionExtrema()
	{
		for(int i = 0; i < this.poblacion.length; i++)
		{
			int random = Utilities.irand(0, 100);

			if(random <= Datos.PROBABILIDAD_MUTACION)
			{
				this.poblacion[i] = new Programa();
			}
		}

		this.poblacionElite = this.selector.guardarElite(this.poblacionElite,
this.poblacion);
	}

	public void cruzar()
	{
		this.poblacion = this.selector.cruzarPoblacion(this.poblacion,
this.poblacionElite);
		this.poblacionElite = this.selector.guardarElite(this.poblacionElite,
this.poblacion);
	}

	@Override
	public String toString()
	{
		StringBuilder sb = new StringBuilder();

		for(int i = 0; i < this.poblacion.length; i++)
		{
			Programa programa = this.poblacion[i];
			double fitness = programa.getFitness();

			sb.append(i).append(". ");
			sb.append("Fitness: ").append(fitness).append(" | ");
			sb.append("Error: ").append(-fitness).append("\n");
		}

		return sb.toString();
	}

	public String toStringElite()
	{
		if(!Utilities.arrayContains(this.poblacionElite, null))
			Arrays.sort(this.poblacionElite, new DescendentComparator<Programa>());

		StringBuilder sb = new StringBuilder();

		for(int i = 0; i < this.poblacionElite.length; i++)
		{
			Programa programa = this.poblacionElite[i];
			double fitness = programa.getFitness();

			sb.append(i).append(". ");
			sb.append("Fitness: ").append(fitness).append(" | ");
			sb.append("Error: ").append(-fitness).append("\n");
		}

		return sb.toString();
	}
}
```

SELECCIONAR OPERACIÓN

```
package ae;

import java.util.Arrays;

import lib_util.log.Errors;
```

```
importLGP.Operacion;

public class Instruccion implements Cloneable
{
	private Operacion operacion;
	private int indiceDestino;
	private int[] indiceOperandos;
	private boolean efectivo;

	public Instruccion(Operacion operacion, int indiceDestino, int... indiceOperandos)
	{
		super();

		this.setOperacion(operacion);
		this.setIndiceDestino(indiceDestino);
		this.setIndiceOperandos(indiceOperandos);
	}

	public Operacion getOperacion()
	{
		return operacion;
	}

	private void setOperacion(Operacion operacion)
	{
		this.operacion = operacion;
	}

	public int getIndiceDestino()
	{
		return indiceDestino;
	}

	private void setIndiceDestino(int registroDestino)
	{
		this.indiceDestino = registroDestino;
	}

	public int[] getIndiceOperandos()
	{
		return indiceOperandos;
	}

	void setIndiceOperandos(int[] indiceOperandos)
	{
		this.indiceOperandos = indiceOperandos;
	}

	public void ejecutarInstruccion(double[] registros)
	{
		Operacion operacion = this.getOperacion();
		int indiceDestino = this.getIndiceDestino();
		int[] indiceOperandos = this.getIndiceOperandos();

		double[] operandos = new double[indiceOperandos.length];

		for(int i = 0; i < operandos.length; i++)
		{
			operandos[i] = registros[indiceOperandos[i]];
		}

		operacion.setOperandos(operandos);

		registros[indiceDestino] = operacion.getResultado();
	}

	public boolean isEfectivo()
	{
		return efectivo;
	}

	public void setEfectivo(boolean efectivo)
	{
		this.efectivo = efectivo;
	}

	@Override
	public String toString()
```

```
	{
		StringBuilder sb = new StringBuilder();

		sb.append("(");
		sb.append(this.getOperacion().getClass().getSimpleName()).append(", ");
		sb.append(this.getIndiceDestino()).append(", ");
		sb.append(Arrays.toString(this.getIndiceOperandos()));
		sb.append(")");

		return sb.toString();
	}

	public Instruccion copiar()
	{
		try
		{
			return this.clone();
		}
		catch(CloneNotSupportedException e)
		{
			Errors.message(e);
			return null;
		}
	}

	@Override
	protected Instruccion clone() throws CloneNotSupportedException
	{
		Instruccion clon = (Instruccion) super.clone();

		clon.operacion = this.operacion.copiar();
		clon.indiceOperandos = this.indiceOperandos.clone();

		return clon;
	}
}
```

SELECTOR RULETA

```
package ae;

import java.util.Arrays;

import lib_util.log.Errors;
import lib_util.util.Utilities;
import datos.Datos;

public class SelectorRuleta implements Selector
{
	private Mutador mutador;
	private Cruzador cruzador;

	public SelectorRuleta(Mutador mutador, Cruzador cruzador)
	{
		this.mutador = mutador;
		this.cruzador = cruzador;
	}

	@Override
	public Programa[] guardarElite(Programa[] poblacionEliteOriginal, Programa[] poblacion)
	{
		Programa[] poblacionElite = poblacionEliteOriginal;

		if(poblacionElite == null)
			poblacionElite = new Programa[Datos.TAM_POBLACION_ELITE];

		for(int i = 0; i < poblacion.length; i++)
		{
			if(Utilities.arrayContains(poblacionElite, poblacion[i])) continue;

			int indicePeorIndividuo = this.getIndicePeorIndividuo(poblacionElite);

			if(poblacion[i].compareTo(poblacionElite[indicePeorIndividuo]) > 0)
			{
				poblacionElite[indicePeorIndividuo] = poblacion[i].copiar();
			}
```

```
		}

		return poblacionElite;
	}

	@Override
	public int getIndicePeorIndividuo(Programa[] poblacion)
	{
		if(poblacion.length < 1) return -1;

		int indicePeorIndividuo = 0;
		Programa peorIndividuo = poblacion[0];

		for(int i = 0; i < poblacion.length; i++)
		{
			if(poblacion[i] == null) return i;

			if(poblacion[i].compareTo(peorIndividuo) < 0)
			{
peorIndividuo.getFitness());
				peorIndividuo = poblacion[i];
				indicePeorIndividuo = i;
			}
		}

		return indicePeorIndividuo;
	}

	@Override
	public Programa[] mutarPoblacion(Programa[] poblacion)
	{
		Programa[] nuevaPoblacion = new Programa[poblacion.length];

		for(int i = 0; i < nuevaPoblacion.length; i++)
		{
			double rand = Utilities.drand(0, 100);

			if(rand < Datos.PROBABILIDAD_MUTACION || rand == 100.0)
			{
				nuevaPoblacion[i] = this.mutador.mutar(poblacion[i]);
			}
			else
			{
				nuevaPoblacion[i] = poblacion[i].copiar();
			}
		}

		return nuevaPoblacion;
	}

	@Override
	public Programa[] cruzarPoblacion(Programa[] poblacion, Programa[] poblacionExtranjera)
	{
		Programa[] poblacionPadre = Utilities.joinArrays(poblacion,
poblacionExtranjera);
		Programa[] nuevaPoblacion = new Programa[poblacion.length];

		int[][] tablaCruce = this.crearTablaCruce(poblacionPadre, poblacion.length);

		for(int i = 0; i < nuevaPoblacion.length; i++)
		{
			nuevaPoblacion[i] =
this.cruzador.cruzar(poblacionPadre[tablaCruce[i][0]], poblacionPadre[tablaCruce[i][1]]);
		}

		return nuevaPoblacion;
	}

	private int[][] crearTablaCruce(Programa[] poblacionPadre, int length)
	{
		int[][] tablaCruce = new int[length][2];

		double[] probabilidades = this.calcularProbabilidades(poblacionPadre);

		for(int f = 0; f < tablaCruce.length; f++)
		{
			for(int c = 0; c < tablaCruce[f].length; c++)
			{
```

```
                                tablaCruce[f][c] = this.sortearIndividuo(poblacionPadre,
probabilidades);
                        }
                }

                return tablaCruce;
        }

        private int sortearIndividuo(Programa[] poblacionPadre, double[] probabilidades)
        {
                double sumaProbabilidades = 0;

                double rand = Utilities.drand(0, 100);

                for(int i = 0; i < poblacionPadre.length; i++)
                {
                        sumaProbabilidades += probabilidades[i];

                        if(rand < sumaProbabilidades) return i;
                }

                Errors.message(Arrays.toString(probabilidades));

                double fitness[] = new double[poblacionPadre.length];

                for(int i = 0; i < fitness.length; i++)
                {
                        fitness[i] = poblacionPadre[i].getFitness();
                }

                Errors.append(Arrays.toString(fitness));


                throw new IllegalArgumentException("Arreglo de probabilidades mal formado");
        }

        private double[] calcularProbabilidades(Programa[] poblacionPadre)
        {
                int cantidad = 0;
                double sumaFitness = 0.0;
                double desplazamiento = 0.0;
                double menorFitness = 0.0;
                double[] probabilidades = new double[poblacionPadre.length];
                boolean menorIniciado = false;


                for(int i = 0; i < probabilidades.length; i++)
                {
                        if(poblacionPadre[i] == null) continue;

                        double fitness = poblacionPadre[i].getFitness();

                        if(Double.isNaN(fitness) || Double.isInfinite(fitness) ||
Double.isInfinite(fitness * multiplicador))
                        {
                                continue;
                        }

                        if(!menorIniciado || fitness < menorFitness)
                        {
                                menorFitness = fitness;
                                menorIniciado = true;
                        }

                        sumaFitness += fitness;
                        cantidad++;

                        if(Double.isNaN(sumaFitness) || Double.isInfinite(sumaFitness))
                        {
                                throw new IllegalStateException("Suma Fitness inválido: " +
sumaFitness);
                        }
                }

                if(menorFitness < 0)
                        desplazamiento = -2 * menorFitness;

                if(Double.isInfinite(sumaFitness + desplazamiento * cantidad))
```

```
		{
			desplazamiento = (Double.MAX_VALUE - ((sumaFitness < 0) ? -sumaFitness :
sumaFitness)) / cantidad;
		}

		sumaFitness += desplazamiento * cantidad;

		if(sumaFitness == 0.0)
		{
			double probabilidad = 100.0 / probabilidades.length;

			for(int i = 0; i < probabilidades.length; i++)
			{
				if(poblacionPadre[i] != null)
				{
Double.isInfinite(poblacionPadre[i].getFitness())) continue;
					probabilidades[i] = probabilidad;
				}
			}
		}
		else
		{
			sumaFitness /= 100.0;

			for(int i = 0; i < probabilidades.length; i++)
			{
				if(poblacionPadre[i] != null)
				{
					double fitness = poblacionPadre[i].getFitness();

					if(Double.isNaN(fitness) || Double.isInfinite(fitness) ||
Double.isInfinite(fitness * multiplicador)) continue;

					probabilidades[i] = (fitness + desplazamiento) /
sumaFitness;
				}
			}
		}

		return probabilidades;
	}
}
```

CALCULO DEL FITNESS

```
package ae;

import java.io.IOException;

import lib_util.files.CSVManager;
import lib_util.log.Errors;
import lib_util.util.Utilities;
import datos.Datos;
import enumeraciones.MetodoFitness;

public class Fitness
{
	public static double calcularFitness(Programa programa)
	{
		int cantidadDatos = 0;
		double errorTotal = 0D;
		double pronosticoAnterior = 1.0;

		CSVManager csv = new
CSVManager(Utilities.getResourceInputStream("ingresos_penales.csv"));

		csv.setSeparator(";");

		try
		{
			csv.readData();
		}
		catch(IOException e)
		{
			Errors.message("Error de lectura de archivo CSV");
			throw new IllegalStateException(e);
```

```
		}

		for(int i = 0; i < csv.size(); i++)
		{
			double[] registros = Datos.obtenerRegistros();



			Double ingresosVerdaderos = Double.valueOf(csv.getData(i, "ingresos"));

			int indiceDato = 0;

			for(int j = 10; j < 14; j++)
			{
				indiceDato = i - (14 - j);

				if(indiceDato >= 0)
					registros[j] = Double.valueOf(csv.getData(indiceDato,
"ingresos"));
				else
					registros[j] = 388;
			}


			if(!Datos.UTILIZAR_HASTA_I_4)
			{
				indiceDato = i - 12;
				if(indiceDato >= 0) registros[10] =
Double.valueOf(csv.getData(indiceDato, "ingresos"));

				indiceDato = i - 13;
				if(indiceDato >= 0) registros[11] =
Double.valueOf(csv.getData(indiceDato, "ingresos"));
			}


			registros[14] = pronosticoAnterior;
			registros[15] = i;

			Double prediccion = programa.getPrediccion(registros);
			Double error = ingresosVerdaderos - prediccion;

			if(Datos.imprimirProceso)
				System.out.println(ingresosVerdaderos + " ; " + prediccion);

			if(Datos.METODO_FITNESS == MetodoFitness.ERROR_CUADRATICO_MEDIO ||
Datos.METODO_FITNESS == MetodoFitness.SUMA_ERROR_CUADRATICO)
			{
				errorTotal += (error * error);
			}
			else if(Datos.METODO_FITNESS == MetodoFitness.ERROR_A_LA_CUARTA_MEDIO)
			{
				errorTotal += (error * error * error * error);
			}
			else if(Datos.METODO_FITNESS == MetodoFitness.ERROR_MEDIO_ABSOLUTO)
			{
				errorTotal += Math.abs(error);
			}
			cantidadDatos++;


			if(Double.isInfinite(errorTotal * 4) || Double.isNaN(errorTotal))
			{

				programa.setFitness(Double.NEGATIVE_INFINITY);
				return Double.NEGATIVE_INFINITY;
			}

			pronosticoAnterior = prediccion;


		}



		if(cantidadDatos == 0)
		{
			throw new IllegalStateException("Cantidad de datos: " + cantidadDatos);
```

```
		}

		if(Datos.METODO_FITNESS == MetodoFitness.ERROR_CUADRATICO_MEDIO ||
Datos.METODO_FITNESS == MetodoFitness.ERROR_MEDIO_ABSOLUTO || Datos.METODO_FITNESS ==
MetodoFitness.ERROR_A_LA_CUARTA_MEDIO)
		{
			errorTotal /= cantidadDatos;
		}

		programa.setFitness(-errorTotal);

		return -errorTotal;
	}
}
```

OPERADOR DE CRUZAMIENTO

```
package ae;

import java.util.List;

import lib_util.util.Factory;
import lib_util.util.Utilities;
import datos.Datos;

public class CruceProgramas implements Cruzador
{
	@Override
	public Programa cruzar(Programa programa1, Programa programa2)
	{
		List<Instruccion> cromosoma1 = programa1.getCromosoma();
		List<Instruccion> cromosoma2 = programa2.getCromosoma();

		List<Instruccion> cromosomaHijo = Factory.getLinkedList();

		int[][] matrizCruce = new int[2][2];

		this.generarMatrizCruce(matrizCruce, cromosoma1.size(), cromosoma2.size());

		cromosomaHijo.addAll(cromosoma1.subList(0, matrizCruce[0][0] + 1));
		if(matrizCruce[1][0] + 1 < matrizCruce[1][1])
			cromosomaHijo.addAll(cromosoma2.subList(matrizCruce[1][0] + 1,
matrizCruce[1][1]));
		cromosomaHijo.addAll(cromosoma1.subList(matrizCruce[0][1], cromosoma1.size()));



		if(cromosomaHijo.size() > Datos.MAX_CANTIDAD_CROMOSOMAS)
		{
			System.out.println("Tam1: " + cromosoma1.size());
			System.out.println("Tam2: " + cromosoma2.size());
			System.out.println("Tam3: " + cromosomaHijo.size());
		}

		return new Programa(cromosomaHijo);
	}

	private void generarMatrizCruce(int[][] matrizCruce, int size1, int size2)
	{
		//FIXME arreglar puntos de cruce
		matrizCruce[0][0] = Utilities.irand(0, size1 - 2);
		matrizCruce[0][1] = Utilities.irand(matrizCruce[0][0] + 1, size1 - 1);
		matrizCruce[1][0] = Utilities.irand(0, size2 - 2);
		matrizCruce[1][1] = Utilities.irand(matrizCruce[1][0] + 1, size2 - 1);

		int segmento1 = matrizCruce[0][0] + 1;
		int segmento2 = matrizCruce[0][1] - matrizCruce[0][0] - 1;
		int segmento3 = size1 - matrizCruce[0][1];
		int segmento5 = matrizCruce[1][1] - matrizCruce[1][0] - 1;

		if((segmento1 + segmento5 + segmento3) > Datos.MAX_CANTIDAD_CROMOSOMAS)
		{
			/*
			System.out.println("------------------------------");
			System.out.println("Size1: " + size1);
			System.out.println("Size2: " + size2);
```

```
                        for(int f = 0; f < matrizCruce.length; f++)
                        {
                                for(int c = 0; c < matrizCruce.length; c++)
                                {
                                        System.out.print("[" + matrizCruce[f][c] + "]");
                                }
                                System.out.println();
                        }

                        System.out.println("segmento5: " + segmento5);
                        /*/
                        matrizCruce[1][1] = matrizCruce[1][0] + segmento2;
                        /*/
                        segmento5 = matrizCruce[1][1] - matrizCruce[1][0] + 1;
                        System.out.println("New size: " + (segmento1 + segmento5 + segmento3));
                        System.out.println("New segmento5: " + segmento5);
                        //*/
                }
        }
}
```

OPERADOR DE MUTACIÓN

```
package ae;

import java.util.List;

import lib_util.util.Utilities;
importLGP.Operacion;
import datos.Datos;

public class MutacionOperador implements Mutador
{
        @Override
        public Programa mutar(Programa programa)
        {
                Programa mutado = programa.copiar();
                int cantidadInstrucciones = programa.getCromosoma().size();

                int cantidadMutaciones = (int)(cantidadInstrucciones *
Datos.INTENSIDAD_MUTACION / 100);

                if(cantidadMutaciones < 1) cantidadMutaciones = 1;

                for(int i = 0; i < cantidadMutaciones; i++)
                {
                        int accion = Utilities.irand(0, 2);

                        if(accion == 0 && (cantidadInstrucciones <
Datos.MAX_CANTIDAD_CROMOSOMAS))
                        {
                                this.agregarInstruccion(mutado);
                        }
                        else if(accion == 1 && cantidadInstrucciones >
Datos.MIN_CANTIDAD_CROMOSOMAS_INICIALES)
                        {
                                this.eliminarInstruccion(mutado);
                        }
                        else
                        {
                                this.modificarIndicesRegistros(mutado);
                        }
                }

                mutado.avisarModificado();

                return mutado;
        }

        private void modificarIndicesRegistros(Programa mutado)
        {
                List<Instruccion> cromosoma = mutado.getCromosoma();
                int modificar = Utilities.irand(0, cromosoma.size() - 1);
```

```
                Instruccion instruccion = cromosoma.get(modificar);
                Operacion operacion = instruccion.getOperacion();

                int[] indiceOperandos = new int[operacion.getCantidadOperandos()];

                for(int i = 0; i < indiceOperandos.length; i++)
                {
                        indiceOperandos[i] = Utilities.irand(0, Datos.CANTIDAD_REGISTROS - 1);
                }

                instruccion.setIndiceOperandos(indiceOperandos);
        }

        private void eliminarInstruccion(Programa mutado)
        {
                List<Instruccion> cromosoma = mutado.getCromosoma();
                int remover = Utilities.irand(0, cromosoma.size() - 1);

                cromosoma.remove(remover);
        }

        private void agregarInstruccion(Programa mutado)
        {
                Operacion operacion = Datos.obtenerOperacion();

                int indiceDestino = Utilities.irand(0, Datos.UMBRAL_CONSTANTES - 1);
                int[] indiceOperandos = new int[operacion.getCantidadOperandos()];

                for(int j = 0; j < indiceOperandos.length; j++)
                {
                        indiceOperandos[j] = Utilities.irand(0, Datos.CANTIDAD_REGISTROS - 1);
                }

                List<Instruccion> cromosoma = mutado.getCromosoma();
                int posicion = Utilities.irand(0, cromosoma.size() - 2);

                cromosoma.add(posicion, new Instruccion(operacion, indiceDestino,
indiceOperandos));
        }
}
```