Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

ELABORACIÓN DE PRONÓSTICOS CON ALGORITMOS GENÉTICOS

Samuel David Orozco y Orozco

Asesorado por el Ing. Freiry Javier Gramajo López

Guatemala, junio de 2007

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ELABORACIÓN DE PRONÓSTICOS CON ALGORITMOS GENÉTICOS

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

SAMUEL DAVID OROZCO Y OROZCO

ASESORADO POR EL ING. FREIRY JAVIER GRAMAJO LÓPEZ

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, JUNIO DE 2007

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paíz Recinos
VOCAL I	Inga. Glenda Patrícia García Soria
VOCAL II	Inga. Alba Maritza Guerrero de López
VOCAL III	Ing. Miguel Ángel Dávila Calderón
VOCAL IV	Br. Kenneth Issur Estrada Ruiz
VOCAL V	Br. Elisa Yazminda Vides Leiva
SECRETARIA	Inga. Marcia Ivónne Véliz Vargas

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	Ing. Freiry Javier Gramajo López
EXAMINADOR	Ing. Edgar René Ornelyz Hoil
EXAMINADOR	Ing. César Augusto Fernández Cáceres
SECRETARIA	Inga. Marcia Ivónne Véliz Vargas

HONORABLE TRIBUNAL EXAMINADOR

Cumpliendo con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

ELABORACIÓN DE PRONÓSTICOS CON ALGORITMOS GENÉTICOS,

tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, en julio de 2005.

Samuel David Orozco y Orozco

DEDICATORIA

Dios	Fuente de la sabiduría, mi fortaleza y padre, quien me dio el regalo de la vida y me permite hoy ver este sueño cumplido.
Mis padres	René Orozco y Eluvia Orozco, por el amor y apoyo que me han brindado siempre. Este es un logro más que me han ayudado a alcanzar.
Mi hermana	Sandra, por estar allí siempre que la necesité y apoyarme durante toda mi carrera.
Mis hermanitas	Karina y Karla, por enseñarme a no olvidar que lo hermoso de la vida está en las cosas sencillas.
Mis abuelos y abuelas	Por esos sabios consejos que sólo la experiencia puede dar.
Mis amigos	De la iglesia, la universidad, el trabajo y la vida. Espero contar siempre con su amistad.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

GLOSARIO

Absoluto Valor numérico sin signo.

Acervo genético Patrimonio o reserva genética de una población compuesto por el grupo completo de alelos únicos que se encontrarían al inspeccionar el material genético de la totalidad de individuos de la población.

Aleatorio Evento que sucede al azar.

Alelo Valor que puede tomar un gen dentro de un cromosoma. Por ejemplo, en la codificación binaria puede ser uno o cero.

Algoritmo Conjunto finito y ordenado de operaciones que permite hallar la solución a un problema.

Aparear Mezclar dos individuos mediante alguna operación genética.

Aptitud Calificación de un individuo para resolver un problema, viene dada por la función de aptitud.

Asimétrico En la operación de cruce, cuando cada parte de los individuos padres se copia de diferente lugar.

Binario	Que puede tomar los valores uno o cero.
Bit	Valor binario uno o cero.
Cíclico	En el análisis de gráficas de series de tiempo cuando se observa un patrón que se repite periódicamente.
Cógidos de gray	Codificación binaria mediante la cual dos números decimales consecutivos difieren únicamente en un bit.
Confiabilidad	Grado en el cual se considera acertado un resultado, es igual a uno menos el porcentaje de error.
Convergencia	Comportamiento estable en los resultados obtenidos por el algoritmo genético, luego de cierto tiempo de ejecución.
Correlación	Relación lineal o no lineal que existe entre dos o más variables.
Cromosoma	Conjunto de genes que representan a un individuo de la población de un algoritmo genético. Cada gen esta codificado de a cuerdo a un método seleccionado. Para el caso de la codificación binaria un cromosoma consiste en una cadena de valores binarios.

Cruce o recombinación	Operación genética que consiste en tomar dos individuos e intercambiar su material genético para formar un nuevo individuo.
Cualitativo	Que no puede ser medido en términos absolutos, en los pronósticos se refiere a aquellos métodos basados en opiniones de expertos.
Cuantitativo	Que puede ser medido y calculado mediante algún método numérico específico.
DAM	Desviación Absoluta de la Media, mide el promedio de las diferencias absolutas entre los valores reales y pronosticados para una variable.
Decodificación	Mecanismo para obtener la información de un individuo que ha sido previamente codificado, se emplea para aplicar la función de aptitud y conocer la calificación de un individuo.
Determinista	Teoría que supone que la evolución de los individuos está, completamente, determinada por las condiciones iniciales.
Distancia de hamming	El número de genes distintos en la misma posición al comparar dos cromosomas.

Diversidad genética	Cuando los individuos de una población son diferentes y presentan resultados diferentes al aplicarles la función de aptitud.
Econometría	Ciencia que aplica técnicas matemáticas y estadísticas a las teorías económicas para su verificación y en la solución de problemas económicos mediante modelos.
Elitismo	Método de selección que tiende a conservar a a los individuos con mejor aptitud en las siguientes generaciones.
EMC	Error Medio Cuadrado, mide el promedio de las diferencias al cuadrado entre valores reales y pronosticados de una variable.
Error o residual de pronóstico	Diferencia entre un valor real y un valor pronosticado para una variable.
Estacionario	Que representa poca o ninguna variabilidad.
Estocástico	Sistema que funciona, sobre todo, por el azar. Esta característica hace que un proceso de resultados diferentes en dos ejecuciones aún cuando se hayan empleado los mismos parámetros para ambas ejecuciones.

Evolución	Proceso continuo de transformación de las especies a través de cambios producidos en sucesivas generaciones. En los algoritmos genéticos el proceso de evolución es introducido mediante las operaciones genéticas.
Fenotipo	Una característica de un individuo.
Gen	Unidad funcional dentro de un cromosoma. Su valor depende de la codificación empleada, para el caso de la codificación binaria un gen viene dado por un uno o un cero.
Genética	Ciencia que estudia como los genes son transmitidos de una generación a otra.
Hacinamiento o apiñamiento	Agrupación estrecha de individuos. Para los algoritmos genéticos el hacinamiento de individuos se basa en la similitud que existe entre ellos mediante algún método de comparación como la distancia de hamming.
JDOM	*Java Document Object Model* (Documento de Modelado de Objetos en Java). Librería de código fuente para manipular datos XML.
Mutación	Alteración realizada en los genes de un cromosoma.

Operador genético	Operador aplicado para la generación de nuevos individuos durante la ejecución de un algoritmo genético.
Óptimo local	Un valor óptimo que puede ser hallado prematuramente en la ejecución de un algoritmo genético, pero que no corresponde a la mejor solución existente para el problema en estudio.
PEMA	Porcentaje de Error Medio Absoluto, mide el promedio de las diferencias absolutas entre los valores reales y pronosticados para una variable como un porcentaje de desviación del valor real.
PME	Porcentaje Medio de Error, mide el promedio de diferencias entre valores reales y pronosticados para una variable como un porcentaje de desviación del valor real.
Polinomio	Expresión matemática en las que constantes y variables se combinan usando adiciones, substracciones y multiplicaciones. Los coeficientes de cada variable son números reales y los exponentes números enteros.
Precisión	Exactitud en el pronóstico de una variable.

Proyección	Estimación del comportamiento futuro de una variable.
Regresión	Análisis acerca de datos pasados que permite encontrar una relación entre dos o más variables.
Selección natural	Selección de individuos basada en el ambiente o naturaleza para favorecer o dificultar la supervivencia de los individuos de una población.
Serie de tiempo	Conjunto de valores para una variable a lo largo de diferentes periodos de tiempo.
Tendencia	Patrón de comportamiento que indica una dirección en los valores en el tiempo para una variable.
XML	*eXtensible Markup Language* –Lenguaje de Marcas Extensible–. Es un metalenguaje que permite definir la gramática de lenguajes específicos.

RESUMEN

El pronóstico es un proceso de estimación de datos futuros mediante la proyección de datos del pasado hacia el futuro, utilizando algún modelo matemático.

Un algoritmo genético es una técnica de programación que imita la evolución natural como estrategia para resolver problemas.

El algoritmo genético diseñado para realizar pronósticos, toma un conjunto de datos históricos como entrada, luego construye una población donde cada individuo es un arreglo de números reales que representan los factores numéricos de un polinomio, se evalúa cada polinomio mediante alguna de las funciones de error de pronóstico que no toman en cuenta el signo del error –Desviación Absoluta de la Media, Error Medio Cuadrado, Porcentaje de Error Medio Absoluto–, se seleccionan los mejores polinomios y se aplica *crossover* y mutación para generar la nueva población. La ejecución finaliza al alcanzar cierto número de ejecuciones o cuando se encuentra un individuo con un error mínimo determinado.

El proceso de experimentación con el algoritmo genético desarrollado inició con un conjunto de datos sencillo donde el modelo matemático que lo representa era un polinomio de grado cero, es decir sólo un término que es una constante. Se emplearon distintas configuraciones de parámetros hasta obtener un conjunto con el cual se encontró el modelo exacto de pronóstico buscado donde el error era igual a cero.

Una vez seleccionada la configuración de parámetros más adecuada, se experimentó con datos reales, sin embargo, con estos datos el error mínimo obtenido fue de 47%. De este experimento se concluyó que debía disminuirse la variabilidad de los datos históricos, sobre todo cuando el comportamiento era cíclico.

Los últimos experimentos probaron distintas configuraciones de parámetros y se obtuvieron mejores resultados al emplear *crossover* de dos puntos asimétrico y selección por hacinamiento determinista. El último experimento empleó estos parámetros con un conjunto de datos reales, pero tomando, únicamente, un ciclo de datos históricos compuesto por doce periodos, esta vez se obtuvo un error del 21%, el cual hace que el modelo aún no sea adecuado para pronosticar.

Se planteó la hipótesis de que los algoritmos genéticos son útiles para construir modelos matemáticos con la estructura de un polinomio que permiten pronosticar valores de una variable en el tiempo y la experimentación determinó que esto no es cierto, pues el algoritmo sólo encontró un modelo adecuado de pronóstico con error de 0 a 1.8% para polinomios de grado tres o inferior. Sin embargo, los datos reales presentan un comportamiento más variable y para estos datos el error mínimo obtenido por el modelo hallado con el algoritmo genético fue de 21%, el cual lo hace inadecuado para pronosticar.

OBJETIVOS

General

Diseñar y desarrollar un algoritmo genético que sea aplicable en el pronóstico de una variable en el tiempo.

Específicos

1. Describir los pronósticos.
2. Describir los algoritmos genéticos.
3. Diseñar un algoritmo genético para realizar pronósticos de una variable.
4. Desarrollar una aplicación que implemente el algoritmo genético diseñado.
5. Evaluar los resultados de los pronósticos obtenidos con la aplicación del algoritmo genético.
6. Realizar experimentos que permitan evaluar el comportamiento del algoritmo genético con distintos parámetros.

HIPÓTESIS

Los algoritmos genéticos son útiles para encontrar modelos matemáticos con la estructura de un polinomio que permitan pronosticar los valores de una variable en el tiempo.

INTRODUCCIÓN

Los algoritmos genéticos comprenden un área de la inteligencia artificial y pueden ser utilizados para resolver una gran cantidad de problemas, mediante el uso de técnicas que imitan los procesos de reproducción y evolución natural tal como selección, cruce y mutación.

Uno de los problemas cuya solución ha sido poco explorada mediante el uso de algoritmos genéticos es el pronóstico de series de tiempo, una serie de tiempo es un conjunto de valores en el tiempo para una variable. Para intentar predecir el comportamiento futuro de una serie de tiempo existen varias técnicas que se basan tanto en opiniones –cualitativas– como en análisis numérico –cuantitativas–, incluso dentro del área de la inteligencia artificial existen soluciones a este problema mediante el uso de redes neuronales.

Los pronósticos constituyen una herramienta importante en la toma de decisiones y se aplican a casos muy variados como las ventas, la producción, el clima o los deportes. Para el caso de las ventas o la producción, un pronóstico acertado permite realizar una mejor planificación y distribución de recursos.

Este trabajo intenta probar que la flexibilidad de los algoritmos genéticos puede permitir, mediante un diseño adecuado, dar solución al problema de pronosticar valores de una variable en el tiempo basándose en un historial de datos. Para ello se ha diseñado y desarrollado una aplicación que implementa un algoritmo genético que toma un conjunto de valores históricos de una variable y genera un modelo matemático que puede ser útil en el pronóstico de valores para dicha variable.

La parte principal de este trabajo se encuentra en el proceso de experimentación con el algoritmo genético desarrollado. Este proceso inicia con varias pruebas de validez de parámetros, en las cuales se observa el comportamiento del algoritmo al emplear distintas configuraciones de parámetros y determinar cuáles son más adecuadas, además de verificar si es necesario realizar algún cambio en la implementación del algoritmo.

Una vez se tienen los parámetros adecuados se procede a analizar historiales de datos hipotéticos y reales para concluir respecto de la utilidad de los algoritmos genéticos en los pronósticos. Se incluye al final de la experimentación un análisis del comportamiento del algoritmo con distintos valores de los parámetros principales y distintos grados de polinomios lo cual permite determinar que parámetros brindan mejores soluciones y hasta que grado de variabilidad en los datos de entrada el algoritmo encuentra una solución.

1. PRONÓSTICOS

1.1. Definición y propósito

“El pronóstico es un proceso de estimación de un acontecimiento futuro proyectando hacia el futuro datos del pasado. Éstos, se combinan sistemáticamente en forma predeterminada para hacer una estimación del futuro.” [1]

“Los pronósticos a menudo son utilizados para poder predecir la demanda del consumidor de productos o servicios, aunque se pueden predecir una amplia gama de sucesos futuros que pudieran de manera potencial influir en el éxito.” [2]

“Pronosticar es el arte y la ciencia de predecir los eventos futuros. Puede involucrar el manejo de datos históricos para proyectarlos al futuro, mediante algún tipo de modelo matemático. Puede ser una predicción del futuro subjetiva o intuitiva. O bien una combinación de ambas, es decir, un modelo matemático ajustado por el buen juicio de un administrador.” [2]

“Es necesario pronosticar debido a que las organizaciones operan en una atmósfera de incertidumbre y, a pesar de este hecho, se deben tomar decisiones que afectan al futuro de la organización. Quienes toman decisiones lo harán mejor a partir del uso adecuado de técnicas de pronóstico.” [1]

1.2. Tipos de pronósticos

Una primera clasificación de los procedimientos de pronóstico es de largo o corto plazo. “Los pronósticos a largo plazo son necesarios para establecer el curso general de la organización para un largo período; de ahí que se conviertan en el enfoque particular de la alta dirección. Los pronósticos a corto plazo se utilizan para diseñar estrategias inmediatas y que usan los administradores de rango medio y de primera línea para enfrentar las necesidades del futuro inmediato.” [1]

“También se podría clasificar a los pronósticos en términos de su posición en el entorno micro-macro, es decir, según el grado en que intervienen pequeños detalles versus grandes valores resumidos.” [1]

“Los procedimientos de pronósticos pueden también clasificarse de acuerdo con su tendencia a ser más cuantitativos o cualitativos. En uno de los extremos, una técnica puramente cualitativa es aquella que no requiere de una abierta manipulación de datos, sólo se utiliza el juicio de quién pronostica. Desde luego, incluso aquí, el juicio del pronosticador es en realidad el resultado de la manipulación mental de datos históricos pasados. En el otro extremo, las técnicas puramente cuantitativas no requieren de elementos de juicio; son procedimientos mecánicos que producen resultados cuantitativos. Por supuesto, ciertos procesos cuantitativos requieren de una manipulación de datos mucho más compleja que otros. Es importante aclarar que junto con los nuevos procedimientos mecánicos y de manipulación de datos, se deben emplear elementos de juicio y sentido común, sólo en esta forma se puede llevar a cabo un pronóstico inteligente.” [1]

La clasificación más utilizada para los pronósticos es la de métodos cualitativos y cuantitativos, a continuación se detalla más cada uno de estos.

1.2.1. Métodos cualitativos

"Éstos se basan principalmente en opiniones de expertos. Su uso es frecuente cuando el tiempo para elaborar el pronóstico es escaso, cuando no se dispone de todos los antecedentes mínimos necesarios o cuando los datos disponibles no son confiables para predecir el comportamiento futuro." [3]

- **Método Delphi.** Se usa para pronósticos a largo plazo, pronósticos de ventas de productos nuevos y pronósticos tecnológicos.
    - *Tiempo estimado*: más de dos meses.
    - *Exactitud*: de regular a muy buena.
- **Investigación de mercados.** Se usa para evaluar y probar hipótesis acerca de mercados reales.
    - *Tiempo estimado:* más de tres meses.
    - *Exactitud:* puede ser excelente, dependiendo del cuidado que se haya puesto en el trabajo.
- **Consenso de un panel.** Tiene los mismos usos que el método Delphi.
    - *Tiempo estimado:* más de dos semanas.
    - *Exactitud:* de baja a regular.
- **Pronósticos visionarios.** Se usa para hacer una profecía del futuro usando la intuición personal.
    - *Tiempo estimado:* una semana.
    - *Exactitud:* mala.
- **Analogía histórica.** Se usa para productos nuevos, basándose en el análisis comparativo de la introducción y crecimiento de productos similares.

- *Tiempo estimado:* más de un mes.
- *Exactitud:* de buena a regular.

1.2.2. Métodos cuantitativos

- **Análisis de series de tiempo.** El análisis consiste en encontrar el patrón del pasado y proyectarlo al futuro. El patrón de una serie de tiempo puede ser:
  - Horizontal o estacionario
  - Tendencia a largo plazo
  - Efecto estacional
  - Efecto cíclico
- **Métodos de proyección.** Estos métodos tratan de encontrar el patrón total de los datos para proyectarlos al futuro, y son:
  - Promedios móviles
  - Suavización exponencial
  - Box-Jenkins
- **Método de separación.** Es aquel que separa la serie en sus componentes para identificar el patrón de cada componente, que se llama, método de descomposición de series de tiempo.
- **Modelos de regresión.** Se basan en la relación lineal entre dos o más variables.
  - Regresión lineal simple
  - Regresión lineal múltiple
- **Modelos econométricos.** Un modelo econométrico es un sistema de ecuaciones de regresión interdependientes que describe algún sector de actividades económicas, ventas o utilidades.

- **Encuestas de intenciones de compra y anticipaciones.** Estas encuestas que se hacen al público y determinan:
    - Las intenciones de compra de ciertos productos.
    - Derivan un índice que mide el sentimiento general sobre el consumo presente y futuro y estiman como afectan estos sentimientos a los hábitos de consumo. Este enfoque para hacer pronósticos es más útil que otras técnicas para seguir el desarrollo de la demanda y para señalar puntos de peligro.
- **Modelo de insumo-producto.** Método de análisis que determina el flujo de bienes y servicios interindustrial o interdepartamental en una economía o en una compañía y su mercado. Muestra flujos de insumos que deben ocurrir para obtener ciertos productos.

1.3. Pasos a seguir en el pronóstico

"Todos los procedimientos formales de pronóstico comprenden la extensión de las experiencias del pasado al futuro incierto. De ahí la suposición de que las condiciones que generaron los datos anteriores son indistinguibles de las condiciones futuras, con excepción de aquellas variables reconocidas de manera explícita por el modelo de pronóstico. Por ejemplo, si se esta pronosticando el índice de desempeño de los empleados en el trabajo, usando sólo como pronóstico la calificación del examen de admisión, se asume que el índice de desempeño en el trabajo de cada persona se afecta sólo por dicho examen. Considerando que la suposición de pasado y futuro indistinguibles no se cumple, resultarán pronósticos imprecisos, a menos que se modifiquen a juicio de quien pronostica." [1]

“La aceptación de que las técnicas de pronósticos funcionan sobre datos generados en sucesos históricos pasados conduce a la identificación de cuatro pasos en el proceso de pronóstico:” [1]

1. Recopilación de datos
2. Reducción o condensación de datos
3. Construcción del modelo
4. Extrapolación del modelo

“El paso 1 sugiere la importancia de obtener datos adecuados y asegurarse que son correctos. Con frecuencia este paso es el mayor reto de todo el proceso de pronóstico y el más difícil de controlar, ya que los pasos siguientes se efectúan sobre los datos, sean o no relevantes para el problema en cuestión. Siempre que se hace necesario obtener datos pertinentes en una organización, abundan los problemas de recopilación y control de calidad.” [1]

“El paso 2, la reducción de datos con frecuencia es necesaria ya que en proceso de pronóstico es posible tener muchos o muy pocos datos. Algunos datos pueden no ser pertinentes al problema, por lo que reducirían la precisión del pronóstico. Otros datos pueden ser los adecuados, pero sólo en ciertos periodos históricos. Por ejemplo, en el pronóstico de ventas de automóviles compactos podría desearse emplear sólo datos de ventas de automóviles a partir del embargo petrolero de la década de 1970, en vez de datos de los últimos 50 años.” [1]

“El paso 3, la construcción del modelo, implica ajustar los datos reunidos en un modelo de pronóstico que sea el adecuado para minimizar el error del pronóstico. Entre más sencillo sea el modelo, será mejor para lograr la aceptación del proceso por parte de los administradores que toman las decisiones en la empresa.” [1]

“Con frecuencia se debe establecer un balance entre un enfoque de pronóstico complejo que ofrezca ligeramente más precisión y un enfoque sencillo que sea fácil de entender y ganar el apoyo de quienes toman las decisiones, de manera que lo utilicen efectivamente. Es obvio que los elementos de juicio forman parte de este proceso de selección.” [1]

“El paso 4 consiste en la extrapolación en sí del modelo de pronóstico, lo cual ocurre una vez que se recolectaron y tal vez redujeron, los datos adecuados y que se seleccionó un modelo de pronóstico apropiado. Es común que quien realizó el pronóstico revise la precisión del proceso mediante el pronóstico de periodos recientes de los que se conocen los valores históricos reales. Es entonces cuando se observan los errores de pronóstico y se resumen de algún modo.” [1]

“Ciertos procedimientos de pronósticos, suman los valores absolutos de los errores y pueden reportar esta suma, o dividirla entre el número de intentos de pronóstico para obtener el error de pronóstico promedio. Otros procedimientos obtienen la suma de cuadrados de los errores, que se compara luego con cifras similares de métodos de pronóstico alternativos. Algunos procedimientos también rastrean y reportan la magnitud de los términos de error sobre el período de pronóstico. El examen de los patrones de error conduce con frecuencia al analista a la modificación del procedimiento de pronóstico, el cual genera después pronósticos más precisos.” [1]

1.4. Medición del error en el pronóstico

“Ya que las técnicas cuantitativas de pronósticos implican, por lo regular, series de tiempo de datos, se desarrolló una notación matemática para hacer referencia a cada período específico.” [1]

“Se empleará la letra Y para denotar una variable de serie de tiempo, a menos que exista más de una variable. El período asociado con una observación se muestra como subíndice. Así, Y_t se refiere al valor de la serie de tiempo en el período t.” [1]

“También se desarrolló una notación matemática para distinguir el valor real de una serie de tiempo y el valor de pronóstico. Se empleará el símbolo ^ (acento circunflejo) sobre un valor, para indicar que se trata de un pronóstico. El valor de pronóstico para Y_t es $\hat{Y}_t$. Con frecuencia se juzga la precisión de una técnica de pronóstico mediante la comparación de la serie original Y_1, Y_2,.... con la serie de pronóstico $\hat{Y}_1$, $\hat{Y}_2$,....” [1]

“La notación básica de pronóstico se resume como:” [1]

Y_t = valor de una serie de tiempo en el período t
$\hat{Y}_t$ = valor del pronóstico para Y_t
$e_t = Y_t - \hat{Y}_t$ = residual o error del pronóstico

“Se han ideado diversos métodos para resumir los errores generados por una técnica particular de pronóstico. La mayoría de estas mediciones implican promediar alguna función de la diferencia entre el valor real y su valor de pronóstico. A menudo se denominan residuales a estas diferencias entre valores observados y los valores de pronóstico.” [1]

“Un residual es la diferencia entre un valor real y su valor de pronóstico. Para calcular el error o residual de cada período de pronóstico, se utilizará la siguiente ecuación:” [1]

$e_t = Y_t - \hat{Y}_t$ = residual o error del pronóstico

Donde

e_t = error del pronóstico en el período t

Y_t = valor real en el período t

$\hat{Y}_t$ = valor del pronóstico en el período t

"Un método para evaluar una técnica de pronóstico consiste en obtener la suma de los errores absolutos. La Desviación Absoluta de la Media (DAM) mide la precisión de un pronóstico mediante el promedio de la magnitud de los errores de pronóstico (valores absolutos de cada error). La DAM resulta de gran utilidad cuando el analista desea medir el error de pronóstico en las mismas unidades de la serie original. La siguiente ecuación muestra como se calcula la DAM:" [1]

$$DAM = \frac{\sum_{t=1}^{n} \left| Y_t - \hat{Y}_t \right|}{n}$$

"Otra técnica para evaluar una técnica de pronóstico es el Error Medio Cuadrado (EMC). Cada error o residual se eleva al cuadrado; luego estos valores se suman y se divide entre el número de observaciones. Este enfoque penaliza los errores mayores de pronósticos, ya que eleva cada uno al cuadrado. Esto es importante pues en ocasiones pudiera ser preferible una técnica que produzca errores moderados a otra que por lo regular tenga errores pequeños, pero que ocasionalmente arroje algunos en extremo grandes. La ecuación para el cálculo del EMC, es la siguiente:" [1]

$$EMC = \frac{\sum_{t=1}^{n} \left(Y_t - \hat{Y}_t \right)^2}{n}$$

“En ocasiones, resulta más útil calcular los errores de pronóstico en términos de porcentaje y no de cantidades. El Porcentaje de Error Medio Absoluto (PEMA) se calcula encontrando el error absoluto en cada período, dividiendo éste entre el valor real observado, para ese período y después promediando estos errores absolutos de porcentaje. Este enfoque es útil cuando el tamaño o magnitud de la variable de pronóstico es importante en la evaluación de la precisión del pronóstico. El PEMA proporciona una indicación de que tan grandes son los errores de pronóstico comparados con los valores reales de la serie. También se puede utilizar el PEMA para comparar la precisión de la misma u otra técnica sobre dos series completamente diferentes. La siguiente ecuación muestra el cálculo del PEMA:” [1]

$$PEMA = \frac{\sum_{t=1}^{n} \frac{\left|Y_t - \hat{Y}_t\right|}{Y_t}}{n}$$

“A veces resulta necesario determinar si un método de pronóstico está sesgado (pronóstico consistentemente alto o bajo). En estos casos, se emplea el Porcentaje Medio de Error (PME), que se calcula encontrando el error en cada período, dividiendo esto entre el valor real de ese período y promediando después estos porcentajes de error. Si un enfoque de pronóstico no está sesgado, la ecuación del PME producirá un porcentaje cercano a cero. Si el resultado es un porcentaje negativo grande, el método de pronóstico está sobrestimado de manera consistente. Si el resultado es un porcentaje positivo grande, el método de pronóstico esta subestimado de forma consistente.” [1]

$$PME = \frac{\sum_{t=1}^{n} \frac{\left(Y_t - \hat{Y}_t\right)}{Y_t}}{n}$$

“Una parte de la decisión para utilizar una técnica de pronóstico en particular es la determinación de si la técnica producirá errores de predicción que se juzguen como suficientemente pequeños. Es en efecto realista esperar que una técnica produzca errores de pronóstico relativamente bajos sobre una base consistente.” [1]

Las cuatro mediciones de precisión de un pronóstico que fueron descritas se utilizan para: [1]

- La comparación de la precisión de dos técnicas diferentes.
- La medición de la utilidad o confiabilidad de una técnica.
- La búsqueda de una técnica óptima.

2. ALGORITMOS GENÉTICOS

2.1. Introducción a los algoritmos genéticos

"Un algoritmo genético (AG) es una técnica de programación que imita a la evolución biológica como estrategia para resolver problemas. Dado un problema específico a resolver, la entrada del AG es un conjunto de soluciones potenciales a ese problema, codificadas de alguna manera, y una métrica llamada función de aptitud que permite evaluar cuantitativamente a cada candidata. Estas candidatas pueden ser soluciones que ya se sabe que funcionan, con el objetivo de que el AG las mejore, pero se suelen generar aleatoriamente." [4]

"Los algoritmos genéticos son algoritmos de búsqueda basados en los mecanismos de la selección natural que combinan la supervivencia de las secuencias mejores adaptadas con cambios aleatorios de información." [5]

"Los algoritmos genéticos pueden verse como una familia de procedimientos de búsqueda adaptativos. Su nombre se deriva de que están basados en modelos de cambio genético en una población de individuos. Esto es:" [6]

- Noción Darwiniana de aptitud (*fitness*) que influye en generaciones futuras.
- Apareamiento que produce descendientes en generaciones futuras.
- Operadores genéticos que determinan la configuración genética de los descendientes.

“Un punto clave de estos modelos, es que el proceso de adaptación no se hace cambiando incrementalmente una sola estructura, sino manteniendo una población de estructuras a partir de las cuales se generan nuevas estructuras usando los operadores genéticos.” [6]

“Cada estructura en la población está asociada con una aptitud y los valores se usan en competencia para determinar qué estructuras serán usadas para formar nuevas estructuras.” [6]

“Una de sus características es su habilidad de explotar información acumulada acerca de un espacio de búsqueda inicialmente desconocido para guiar la búsqueda subsecuente a subespacios útiles.” [6]

“Su aplicación está enfocada sobre todo a espacios de búsqueda grandes, complejos y poco entendidos.” [6]

“El precio es que se pueden necesitar un número grande de muestras para que se tenga suficiente información para guiar muestras subsecuentes a subespacios útiles.” [6]

“En su forma más simple, un algoritmo genéticos está orientado a desempeño (por ejemplo, hacer cambios estructurales para mejorar el desempeño).” [6]

“Una de las ideas más importantes es definir estructuras admisibles en el sentido que estén bien definidas y puedan ser evaluadas.” [6]

Los algoritmos genéticos difieren con los métodos tradicionales de búsqueda y optimización en los siguientes aspectos: [6]

- Trabajan con un conjunto de parámetros codificados y no con los parámetros mismos.
- Inician la búsqueda desde un conjunto de puntos, no de uno solo.
- Usan una función a optimizar en lugar de la derivada u otro conocimiento adicional.
- Usan reglas de transición probabilísticas no determinísticas.

2.2. Anatomía de un algoritmo genético

Un algoritmo genético esta compuesto por: [6]

- **Módulo evolutivo:** mecanismo de decodificación (interpreta la información de un cromosoma) y función de evaluación (mide la calidad del cromosoma). Sólo aquí existe información del dominio.
- **Módulo poblacional:** tiene una representación poblacional y técnicas para manipularla (técnica de representación, técnica de arranque, criterio de selección y de reemplazo). Aquí también se define el tamaño de la población y la condición de terminación.
- **Módulo reproductivo:** contiene los operadores genéticos.

2.3. Características de los algoritmos genéticos

Algunas de las características de los algoritmos genéticos son: [7]

- Son algoritmos estocásticos. Dos ejecuciones distintas pueden dar dos soluciones distintas.

- Son algoritmos de búsqueda múltiple, por lo tanto dan varias soluciones. Aunque habitualmente las energías de los individuos de la población final es similar, los individuos suelen ser distintos entre si.
- Son los algoritmos que hacen una barrida amplia al subespacio de posibles soluciones válidas, y con diferencia. De hecho, se considera que, de todos los algoritmos de optimización estocásticos, los algoritmos genéticos son de los más exploratorios disponibles.
- La convergencia del algoritmo es poco sensible a la población inicial si esta se escoge de forma aleatoria y es lo suficientemente grande.
- Por su grado de penetración casi nulo, la curva de convergencia asociada al algoritmo presenta una convergencia excepcionalmente rápida al principio, que casi enseguida se bloquea. Esto de debe a que el algoritmo genético es excelente descartando subespacios realmente malos. Cada cierto tiempo, la población vuelve dar el salto evolutivo, y se produce un incremento en la velocidad de convergencia excepcional. La razón de esto es que algunas veces aparece una mutación altamente beneficiosa, o un individuo excepcional, que propaga algún conjunto de cromosomas excepcional al resto de la población.
- La optimización es función de la representación de los datos. Una buena codificación puede hacer la programación y resolución muy sencillas, mientras que una codificación errada obligará a estudiar que los individuos cumplan las restricciones del problema. Además, la velocidad de convergencia va a estar fuertemente influenciada por la representación.
- Es una búsqueda paramétricamente robusta. Eso quiere decir que los parámetros del algoritmo escogidos deben ser realmente malos para que no converja.

- Los algoritmos genéticos son intrínsecamente paralelos. Esto significa que, independientemente de que se hayan implementado de forma paralela o no, buscan en distintos puntos del espacio de soluciones de forma paralela.

2.4. Analogía de un algoritmo genético con la naturaleza

"La idea básica de un algoritmo genético es la siguiente: generar un conjunto con algunas de las posibles soluciones. Cada una va a ser llamada individuo, y a dicho conjunto se le denominará población." [7]

"Cada individuo tiene una información asociada a él. En un problema de optimización corresponde a las variables libres, es decir, aquellas a las que el algoritmo tiene que asignar un valor para que una función sea mínima o máxima para esos valores. Esa función es la denominada función de adaptación y determina el grado de adaptación de un individuo. A dicha información se la va a denominar código genético." [7]

"Las características de los individuos, sean beneficiosas o no, se van a denominar fenotipos. La información asociada a un individuo se compone de partes indivisibles denominados cromosomas." [7]

"Un fenotipo puede estar en más de un cromosoma, en cuyo caso puede ser que el hijo herede un fenotipo que no tenía ni el padre ni la madre, sino una combinación de ambos. Un ejemplo en el humano es el color de la piel o la estructura del cráneo. En caso de que el hijo tenga parte de los genes del padre y parte de los genes de la madre que intervienen en un fenotipo, se va a crear una característica nueva asociada a ese fenotipo." [7]

Este no es un enfoque muy frecuente, ya que debemos asegurar que el conjunto de los fenomas tendrá ley de composición interna respecto al operador de cruce definido sobre el alfabeto cromosómico. Por otro lado, que un cromosoma codifique más de un fenotipo es más raro todavía. El cromosoma debe tener en dicho caso tantos valores como el producto del número de valores posibles que tenga cada fenotipo del cromosoma.

“Es de vital importancia la forma de codificar los fenotipos en los cromosomas y la determinación de qué es fenotipo, es decir, como la información va a ser almacenada en el código genético. Escoger equivocadamente la forma de almacenar la información puede ralentizar la convergencia, es decir, que se tardará más en encontrar la solución ó nunca convergerá en una solución debido a que la población esta errando aleatoriamente por efecto de las mutaciones y de los cruzamientos sin llegar nunca a un punto estable, a este fenómeno se le denomina deriva genética.” [7]

2.5. Codificación de las variables

"Los algoritmos genéticos requieren que el conjunto se codifique en un cromosoma. Cada cromosoma tiene varios genes, que corresponden a sendos parámetros del problema. Para poder trabajar con estos genes en el ordenador, es necesario codificarlos en una cadena, es decir, una ristra de símbolos (números o letras) que generalmente va a estar compuesta de ceros y unos.” [8]

Los tres métodos principales de codificación de variables son: [4]

- Cadenas binarias, secuencias de unos y ceros donde el dígito de cada posición representa el valor de algún aspecto de la solución.

- Cadenas de enteros o números decimales donde cada posición representa algún aspecto particular de la solución. Este método permite una mayor precisión y complejidad que el método comparativamente restringido de utilizar sólo números binarios, y a menudo está intuitivamente más cerca del espacio de problemas.
- Cadenas de letras donde cada letra representa un aspecto específico de la solución.

El método elegido para codificar las variables debe facilitar la definición de operadores que causen los cambios aleatorios en los individuos seleccionados: cambiar un 0 por un 1 o viceversa, sumar o restar al valor de un número una cantidad elegida al azar, o cambiar una letra por otra.

“Otra estrategia, desarrollada principalmente por John Koza, de la Universidad de Stanford, y denominada programación genética, representa a los programas como estructuras de datos ramificadas llamadas árboles. En este método, los cambios aleatorios pueden generarse cambiado el operador o alterando el valor de un cierto nodo del árbol, o sustituyendo un subárbol por otro.” [4]

“La elección de la codificación de las variables no es trivial, especialmente si estamos tratando un problema de optimización. Por ejemplo, una codificación directa de los números enteros puede dar problemas a la hora de que el algoritmo converja, ya que números consecutivos, como por ejemplo, el 15 y el 16, al pasarlos a binario son muy diferentes (10000, 01111) con lo que una solución con el valor 15 en un campo, difícilmente llegará a evolucionar a una solución con el valor 16 (ya que debería cambiar simultáneamente todos los bits). A este problema se le conoce como Picos de Hamming." [9]

Para disminuir el efecto de los “Picos de Hamming” se pueden utilizar otras codificaciones. Por ejemplo, codificando según los "Códigos de Gray" asegura que enteros consecutivos solo se diferencien en un bit.

2.6. Evaluación y selección

“Durante la evaluación se decodifica el gen y se convierte en una serie de parámetros de un problema. Luego se halla la solución del problema a partir de esos parámetros y se le da una puntuación en función de lo cerca que esté de la mejor solución. A esta puntuación se le llama *fitness*.” [8]

El siguiente ejemplo [8] ilustra el proceso de evaluación de un algoritmo genético. Se quiere hallar el máximo de una función cuya gráfica es una parábola invertida con el máximo en $x = 1$. El único parámetro del problema es la variable x. La optimización consiste en hallar un x tal que $F(x)$ sea máximo. Se crea entonces una población de cromosomas, cada uno de los cuales contiene una codificación binaria del parámetro x. Cada byte, cuyo valor está comprendido entre 0 y 255, se transformará para ajustarse al intervalo [-1,1], donde el objetivo es hallar el máximo de la función. La tabla I muestra los resultados obtenidos para cuatro cromosomas.

Tabla I. Ejemplo de resultados obtenidos con una función de evaluación

Valor binario	Decodificación	Evaluación f(*x*)
10010100	21	0,9559
10010001	19	0,9639
101001	-86	0,2604
1000101	-58	0,6636

El fitness determina siempre los cromosomas que se van a reproducir, y aquellos que se van a eliminar.

2.6.1. Métodos de selección

Entre las técnicas que un algoritmo genético puede utilizar para seleccionar a los individuos que deben copiarse hacia la siguiente generación están: [4]

- **Selección elitista:** se garantiza la selección de los miembros más aptos de cada generación. La mayoría de los algoritmos genéticos no utilizan elitismo puro, sino que usan una forma modificada por la que el individuo mejor, o algunos de los mejores, son copiados hacia la siguiente generación en caso de que no surja nada mejor.
- **Selección proporcional a la aptitud:** los individuos más aptos tienen más probabilidad de ser seleccionados, pero no la certeza.
- **Selección por rueda de ruleta:** una forma de selección proporcional a la aptitud en la que la probabilidad de que un individuo sea seleccionado es proporcional a la diferencia entre su aptitud y la de sus competidores.
- **Selección escalada:** al incrementarse la aptitud media de la población, la fuerza de la presión selectiva también aumenta y la función de aptitud se hace más discriminadora. Este método puede ser útil para seleccionar más tarde, cuando todos los individuos tengan una aptitud relativamente alta y sólo les distingan pequeñas diferencias en la aptitud.
- **Selección por torneo:** se eligen subgrupos de individuos de la población, y los miembros de cada subgrupo compiten entre ellos. Sólo se elige a un individuo de cada subgrupo para la reproducción.
- **Selección por estado estacionario:** la descendencia de los individuos seleccionados en cada generación vuelven al acervo genético preexistente, reemplazando a algunos de los miembros menos aptos de la siguiente generación. Se conservan algunos individuos entre generaciones.

- **Selección por rango:** a cada individuo de la población se le asigna un rango numérico basado en su aptitud, y la selección se basa en este rango, en lugar de las diferencias absolutas en aptitud. La ventaja de este método es que puede evitar que individuos muy aptos ganen dominancia al principio a expensas de los menos aptos, lo que reduciría la diversidad genética de la población y podría obstaculizar la búsqueda de una solución aceptable.
- **Selección generacional:** la descendencia de los individuos seleccionados en cada generación se convierte en toda la siguiente generación. No se conservan individuos entre las generaciones.
- **Selección jerárquica:** los individuos atraviesan múltiples rondas de selección en cada generación. Las evaluaciones de los primeros niveles son más rápidas y menos discriminatorias, mientras que los que sobreviven hasta niveles más altos son evaluados más rigurosamente. La ventaja de este método es que reduce el tiempo total de cálculo al utilizar una evaluación más rápida y menos selectiva para eliminar a la mayoría de los individuos que se muestran poco o nada prometedores, y sometiendo a una evaluación de aptitud más rigurosa y computacionalmente más costosa sólo a los que sobreviven a esta prueba inicial.

Algunos de estos métodos son mutuamente exclusivos, pero otros pueden utilizarse en combinación, algo que se hace a menudo.

2.6.1.1 Hacinamiento determinista

"El hacinamiento determinista ó DC por sus siglas en inglés (*Deterministic Crowding*) se basa en el principio de competición estricta. Cualquier individuo puede ser recombinado con otro pero únicamente el hijo que más se asemeje al padre puede reemplazarlo." [10]

De esta manera los individuos tienden a competir entre sí con otros que tienen el mismo punteo obteniendo varios óptimos locales. Esta característica es útil para prevenir una convergencia prematura ya que explora diferentes espacios en paralelo.

Otra ventaja es que existe un mecanismo elitista inherente en el método. Los algoritmos genéticos que no tienen una buena diversidad genética convergen rápidamente en un óptimo local o no pueden mantener un óptimo global durante la evolución.

Un algoritmo genético con hacinamiento determinista estándar trabaja de la siguiente manera: [10]

1. Inicializar la población aleatoriamente.
2. Aleatoriamente formar parejas de individuos.
3. Después de la recombinación, dos padres producen dos hijos.
4. De acuerdo a una regla de paridad, cada hijo es comparado con un padre.
5. Si el punteo del hijo es superior que el del padre, el padre es reemplazado.
6. Ir al paso 2 hasta que se satisfaga el criterio de parada.

Cada hijo es comparado con el padre con que tiene mayor similitud. Si los dos padres son idénticos, cada hijo es comparado con un padre. Si los dos hijos son más similares a un padre y ambos punteos son superiores a los del padre, el hijo más similar al padre gana.

Para medir la similitud entre un hijo y un padre se emplea la distancia de Hamming. “Si dos listas de elementos son comparadas, la distancia de Hamming esta dada por el número de elementos que no son iguales”. [11]

2.7. Funcionamiento de un algoritmo genético

Luego de haber codificado las variables y haber definido el método de selección, el algoritmo genético procede de la siguiente manera: [8]

1. Se generan aleatoriamente una serie de cromosomas.
2. Se evalúa la puntuación de cada uno de los genes.
3. Se permite a cada uno de los individuos reproducirse de a cuerdo con su puntuación.
4. Se empareja a los individuos de la nueva población, haciendo que intercambien material genético y que alguno de los bits de un gen se vea alterado debido a una mutación espontánea.

Un algoritmo genético también tiene parámetros que se tienen que fijar para cada ejecución: [8]

- **Tamaño de la población:** debe de ser suficiente para garantizar la diversidad de las soluciones y tiene que crecer más o menos con el número de bits del cromosoma.
- **Condición de terminación:** lo más habitual es que la condición de terminación sea la convergencia del algoritmo genético o un número prefijado de generaciones.

El procedimiento que sigue un algoritmo genético puede resumirse en el algoritmo de la figura 1.

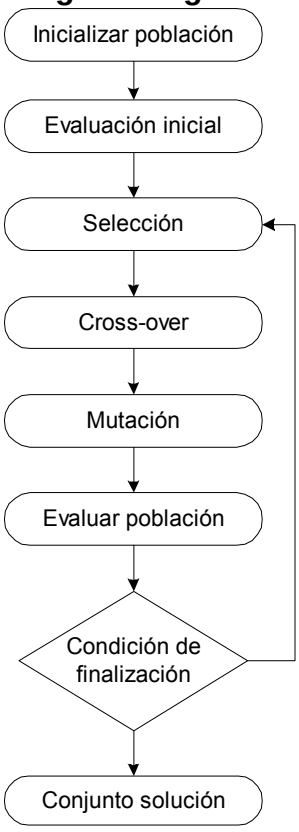
Un posible esquema para representar la implementación de un algoritmo genético se muestra en la figura 2.

Figura 1. Procedimiento de un algoritmo genético

```
Procedimiento AG
Tiempo = 0
Inicializa Población (tiempo)
Evalúa Población (tiempo)
Mientras no condición de terminación
      Tiempo = tiempo + 1
      Construye Población  (tiempo)
      a partir de población (tiempo – 1) usando:
      o  Selección
      o  Modifica Población (tiempo) usando Operadores Genéticos
      o  Evalúa Población (tiempo)
      o  Reemplaza
Fin Mientras
```

Fuente: Eduardo F. Morales. **Búsqueda optimización y aprendizaje.**

Figura 2. Diagrama de funcionamiento de una posible implementación de algoritmo genético



Fuente: Jorge Martín y Diego García. **Seminario: algoritmos genéticos.** Pág. 8

2.8. Operadores genéticos

2.8.1. *Crossover*

"Es el principal operador genético y consiste en el intercambio de material genético entre dos cromosomas." [8]

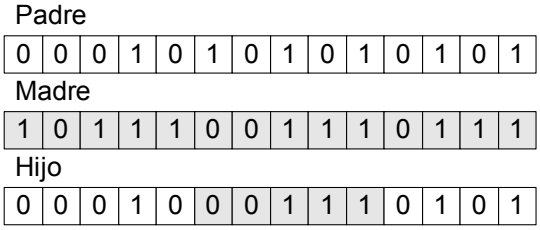
"Para aplicar el *crossover*, entrecruzamiento o recombinación, se escogen aleatoriamente dos miembros de la población. No pasa nada si se emparejan dos descendientes de los mismos padres, ello garantiza la perpetuación de un individuo con buena puntuación. Sin embargo, si esto sucede demasiado a menudo, puede crear problemas: toda la población puede aparecer dominada por los descendientes de algún gen que puede tener caracteres no deseados. Esto se suele denominar en otros métodos de optimización atranque en un mínimo local, y es uno de los principales problemas con los que se enfrentan los que aplican algoritmos genéticos." [8]

"El *crossover* se usa en conjunto con el teorema de los esquemas el cual se basa en la noción de bloques de construcción, así una buena solución a un problema está constituida por buenos bloques. El *crossover* es el encargado de mezclar bloques buenos que se encuentren en los diversos progenitores. La presión selectiva se encarga de que sólo los buenos bloques se perpetúen y poco a poco vayan formando una buena solución. El teorema de los esquemas viene a decir que la cantidad de buenos bloques se va incrementando con el tiempo de ejecución de un algoritmo genético, y es el resultado teórico más importante en algoritmos genéticos." [8]

El intercambio genético se puede llevar a cabo de muchas formas: [8]

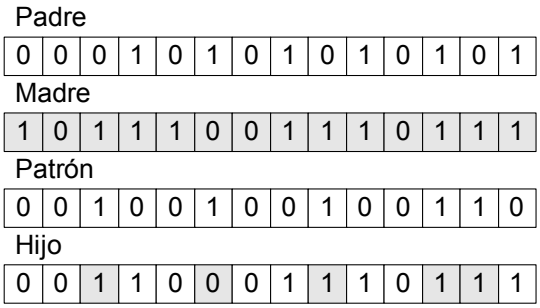
- ***Crossover* de n-puntos:** los dos cromosomas se cortan por n puntos, y el material genético situado entre ellos se intercambia. Lo más habitual es un *crossover* de un punto o de dos puntos. Un ejemplo de *crossover* de dos puntos se muestra en la figura 3.

Figura 3. *Crossover* de n puntos



- ***Crossover* uniforme:** se genera un patrón aleatorio de 1s y 0s, y se intercambian los bits de los dos cromosomas que coincidan donde hay un 1 en el patrón. O bien se genera un número aleatorio para cada bit, y si supera una determinada probabilidad se intercambia ese bit entre los dos cromosomas. Un ejemplo de *crossover* uniforme se muestra en la figura 4, donde se ha tomado un bit del padre cuando el valor del patrón es cero y un bit de la madre cuando el valor del patrón es uno.

Figura 4. *Crossover* uniforme



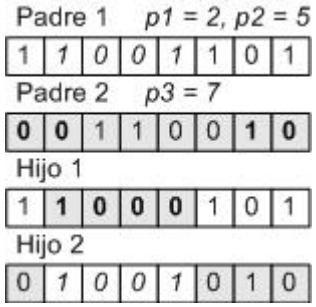
- ***Crossover* especializados:** en algunos problemas, aplicar aleatoriamente el *crossover* da lugar a cromosomas que codifican soluciones inválidas, en este caso hay que aplicar el *crossover* de forma que genere siempre soluciones válidas.

Existe una extensión del *crossover* de dos puntos en el cual los puntos de cruce pueden ser diferentes en los padres, este es el *crossover* asimétrico de dos puntos ó ATC por sus siglas en inglés (*asymmetric two-point crossover*). El ATC funciona de la siguiente manera: [10]

- Seleccionar dos puntos de cruce *p*1 y *p*2 en el primer padre.
- Seleccionar un punto de cruce *p*3 en el segundo padre.
- Crear el primer hijo: reemplazar los genes entre *p*1 y *p*2 en el primer padre con los genes del segundo padre iniciando en *p*3.
- Crear el segundo hijo: reemplazar los genes entre *p*1 y *p*2 en el segundo padre con aquellos entre *p*1 y *p*2 en el primer padre.

Cada individuo padre es tratado como una lista circular de genes, como se muestra en la figura 5 al llegar al último gen en el segundo padre se continúa en el inicio.

Figura 5. *Crossover* de dos puntos asimétrico



2.8.2. Mutación

"En la evolución una mutación es un suceso poco común (sucede aproximadamente una de cada mil replicaciones). En la mayoría de los casos las mutaciones son letales, pero en promedio contribuyen a la diversidad genética de la especie. En un algoritmo genético tendrán el mismo papel y la misma frecuencia (muy baja)." [8]

"Una vez establecida la frecuencia de mutación, por ejemplo, uno por mil, se examina cada bit de cada cadena cuando se vaya a crear el nuevo individuo a partir de sus padres (normalmente de forma simultánea al *crossover*), si un número generado aleatoriamente está por debajo de la frecuencia de mutación, se cambiará el bit (de 0 a 1 o de 1 a 0), si no se dejará como está. Dependiendo del número de individuos que haya y del número de bits por individuo, puede resultar que las mutaciones sean extremadamente raras en una sola generación." [8]

"No conviene abusar de la mutación, es cierto que es un mecanismo generador de diversidad y por tanto, la solución cuando un algoritmo genético está estancado, pero también es cierto que reduce el algoritmo genético a una búsqueda aleatoria. Siempre es más conveniente usar otros mecanismos de generación de diversidad, como aumentar el tamaño de la población o garantizar la aleatoriedad de la población inicial." [8]

"El operador mutación junto con el *crossover* y el método de selección de ruleta constituyen un algoritmo genético simple, SGA, introducido por Goldberg en su libro." [8]

2.8.3. Otros operadores

"No se usan en todos los problemas, sino sólo en algunos, y en principio su variedad es infinita. Generalmente son operadores que exploran el espacio de soluciones de una forma más ordenada, y que actúan más en las últimas fases de la búsqueda, en la cual se pasa de soluciones casi buenas a buenas soluciones." [8]

2.8.3.1 Cromosomas de longitud variable

"Hay problemas en los que no se conoce de antemano el número de parámetros de un problema. Por ejemplo, en un problema de clasificación, donde dado un vector de entrada queremos agruparlo en una serie de clases, no se conoce cuantas clases hay. O en diseño de redes neuronales, puede que no se sepa (de hecho, nunca se sabe) cuántas neuronas se van a necesitar." [8]

"En estos casos, necesitamos dos operadores más: añadir y eliminar. Estos operadores se utilizan para añadir un gen, o eliminar un gen del cromosoma. La forma más habitual de añadir un gen es duplicar uno ya existente, el cual sufre mutación y se añade al lado del anterior. En este caso, los operadores del algoritmo genético simple (selección, mutación, *crossover*) funcionarán de la forma habitual, salvo que sólo se hará *crossover* en la zona del cromosoma de menor longitud." [8]

"Estos operadores permiten, además, crear un algoritmo genético de dos niveles: a nivel de cromosoma y a nivel de gen." [8]

“Supongamos que, en un problema de clasificación, hay un gen por clase. Se puede asignar una puntuación a cada gen en función del número de muestras que haya clasificado correctamente. Al aplicar estos operadores, se duplicarán los alelos con mayor puntuación, y se eliminarán aquellos que hayan obtenido menor puntuación, o cuya puntuación sea nula.” [8]

2.8.3.2 Operadores de nicho (ecológico)

“Estos operadores están encaminados a mantener la diversidad genética de la población, de forma que cromosomas similares sustituyan sólo a cromosomas similares, y son especialmente útiles en problemas con muchas soluciones; un algoritmo genético con estos operadores es capaz de hallar todos los máximos, dedicándose cada especie a un máximo. Más que operadores genéticos, son formas de enfocar la selección y la evaluación de la población.” [8]

“Una de las formas de llevar esto a cabo ya ha sido nombrada, la introducción del *crowding* (hacinamiento). Otra forma es introducir una función de compartición o *sharing*, que indica cuán similar es un cromosoma al resto de la población. La puntuación de cada individuo se dividirá por esta función de compartición, de forma que se facilita la diversidad genética y la aparición de individuos diferentes.” [8]

“También se pueden restringir los emparejamientos, por ejemplo, a aquellos cromosomas que sean similares. Para evitar las malas consecuencias del *inbreeding* que suele aparecer en poblaciones pequeñas, estos periodos se intercalan con otros periodos en los cuales el emparejamiento es libre.” [8]

2.8.3.3 Operadores especializados

"En una serie de problemas hay que restringir las nuevas soluciones generadas por los operadores genéticos, pues no todas las soluciones generadas van a ser válidas, sobre todo en los problemas con restricciones. Por ello, se aplican operadores que mantengan la estructura del problema. Otros operadores son simplemente generadores de diversidad, pero la generan de una forma determinada:" [8]

- **Zap**: en vez de cambiar un solo bit de un cromosoma, cambia un gen completo de un cromosoma.
- **Creep:** este operador aumenta o disminuye en uno el valor de un gen, sirve para cambiar suavemente y de forma controlada los valores de los genes.
- **Transposición:** similar al crossover y a la recombinación genética, pero dentro de un solo cromosoma, dos genes intercambian sus valores sin afectar al resto del cromosoma. Similar a este es el operador de eliminación-reinserción, en el que un gen cambia de posición con respecto a los demás.

2.9. Ejemplo simple de un algoritmo genético

Para comprender mejor el funcionamiento de un algoritmo genético, se presenta en esta sección un ejemplo sencillo. Este ejemplo ha sido tomado de [12].

El ejemplo consiste en maximizar la siguiente función:

$f(x) = x^2$

Se desea encontrar el valor de *x* que maximice el valor de *f*(*x*), restringiendo a la variable *x* a tomar valores enteros comprendidos entre 0 y 31. Es obvio que el valor máximo de *f*(*x*) se obtiene para *x*=31, el cual es de 961. Sin embargo, se ha elegido este problema por su sencillez para comprender el funcionamiento de los algoritmos genéticos.

Lo primero que debe hacerse es encontrar una manera de codificar las posibles soluciones (posibles valores de *x*). Se utilizará una codificación binaria, con la cual es necesario tener individuos de longitud igual a cinco (5 bits), como se muestra en la tabla II.

Tabla II. Codificación de ejemplo para un algoritmo genético

X	**Valor codificado (individuo)**
1	00001
2	00010
3	00011
4	00100
…	…
28	11100
29	11101
30	11110
31	11111

La población de un algoritmo genético esta constituida por un conjunto de individuos. Para éste ejemplo se tomará una población de seis individuos.

El algoritmo genético debe partir de una población inicial. Ésta población se forma generando individuos (cadenas de cinco bits) al azar. Para el ejemplo, se partirá de la población inicial mostrada en la tabla III.

Tabla III. Población inicial de ejemplo para un algoritmo genético

Instancia	Individuo	*X*	*F*(*x*)
1	01100	12	144
2	10010	18	324
3	01111	15	225
4	11000	24	576
5	11010	26	676
6	00001	1	1
			1946

Luego debe realizarse un proceso de selección para obtener a los mejores individuos de la población. Para ello es necesario contar con una función de evaluación que indique cuáles son los mejores individuos. Para el ejemplo, la función de evaluación esta dada por la función que se desea maximizar $f(x) = x^2$, y serán mejores aquellos individuos que den un mayor valor de $f(x)$. Para la población inicial, el mejor individuo es el $x = 26$ con $f = 676$. Y la frecuencia media es 324.33.

Una manera de realizar el proceso de selección es mediante torneo entre dos. A cada individuo de la población se le asigna una pareja aleatoriamente y entre ellos se establece un torneo: el mejor genera dos copias y el peor se desecha. La tabla IV muestra este proceso para la población inicial del ejemplo.

Tabla IV. Proceso de selección de ejemplo

Instancia	Individuo	*x*	*f*(*x*)	Pareja asignada	Mejor instancia	Nueva población
1	01100	12	144	6	1	01100
2	10010	18	324	3	2	10010
3	01111	15	225	2	2	10010
4	11000	24	576	5	5	11010
5	11010	26	676	4	5	11010
6	00001	1	1	1	1	01100

Luego de realizar la selección, se aplican los operadores genéticos. Para el ejemplo se aplicará un cruce (*crossover*) de un punto (1x). El proceso consiste en formar parejas entre los individuos aleatoriamente al igual que para la selección.

Dados dos individuos pareja (padre y madre) se establece un punto de cruce aleatorio, que no es más que un número aleatorio entre 1 y 4 (la longitud del individuo menos uno). La operación de cruce consiste en generar dos nuevos individuos intercambiando los bits de los padres en el punto de cruce. Así, si se obtiene un punto de cruce igual a 3 significa que el un hijo tomará los primeros 3 bits del padre y los últimos 2 bits de la madre, mientras que el otro hijo tomará los primeros 3 bits de la madre y los últimos 2 bits del padre.

El proceso de cruce para la población del ejemplo se muestra en las tablas V y VI.

Tabla V. Ejemplo de preparación para el cruce

Instancia	Población actual	Pareja asignada
1	01100	5
2	10010	3
3	10010	2
4	11010	6
5	11010	1
6	01100	4

Tabla VI. Ejemplo de operación de cruce

Pareja (padre-madre)	Punto de cruce	Padre	Madre	Población resultante	*x*	*f*(*x*)
1-5	1	**01100**	11010	**0**1010	10	100
				1**1100**	28	784
2-3	3	**10010**	10010	**100**10	18	324
				100**10**	18	324
4-6	4	**11010**	01100	**1101**0	26	676
				0110**0**	12	144
						2352

Ahora, en la nueva población el mejor individuo es $x = 28$ con $f = 784$. Y la frecuencia media es de 392. Esto indica que luego de la selección y el cruce los individuos de la población han mejorado.

El siguiente paso consiste en realizar nuevamente la selección y el cruce, tomando como entrada la población obtenida en la iteración anterior. Este proceso se repite hasta obtener un individuo con cierta puntuación o hasta alcanzar un número de iteraciones determinado.

Generalmente, en los problemas donde se aplican los algoritmos genéticos existe la tendencia a la homogeneización de la población, es decir que todos los individuos sean idénticos. Esto impide que el algoritmo siga explorando nuevas soluciones, con lo que se puede quedar estancado en un máximo local no muy bueno.

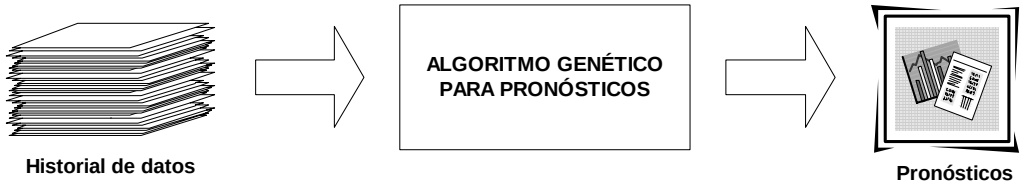
Existen varias técnicas para evitar esta “deriva genética”. El mecanismo más elemental, aunque no lo suficientemente eficaz, es introducir una mutación luego de la selección y el cruce. La mutación consiste en elegir individuos al azar de la población y alterar alguno de sus genes. Para el ejemplo esto se haría cambiando un bit al azar de 0 a 1 (ó de 1 a 0) de uno o más individuos de la población.

3. DISEÑO DEL ALGORITMO GENÉTICO PARA PRONÓSTICOS

3.1. Una primera aproximación

Como una primera aproximación del algoritmo genético para realizar pronósticos se puede pensar en un algoritmo que pronostique directamente el valor de la variable, es decir, un algoritmo que reciba como entrada los valores anteriores o históricos de la variable a predecir y que devuelva el pronóstico para un período determinado. Esto se muestra en la figura 6.

Figura 6. Primera aproximación al algoritmo genético para realizar pronósticos



De esta manera, en la construcción del modelo que constituye el tercer paso para la realización de un pronóstico (ver sección 1.3), el algoritmo genético a desarrollar sería el modelo que se ajusta a los datos y que se utilizará para pronosticar valores no conocidos.

Aunque inicialmente ésta parece ser una solución factible, no lo es. A continuación se explicará porqué no puede utilizarse un algoritmo genético para realizar pronósticos directamente.

En primer lugar, una de las características de los algoritmos genéticos es que son estocásticos (ver sección 2.3) y por tanto dos ejecuciones del algoritmo darán resultados completamente distintos.

Un proceso estocástico es una sucesión de eventos que se desarrolla en el tiempo y cuyo resultado en cualquier etapa contiene algún elemento que depende del azar. De manera más formal, un proceso estocástico se define como una colección indexada de variables aleatorias (X_t) donde el índice "t" toma valores de un conjunto "T" dado. [13]

Esto representa un problema debido a que se busca predecir el valor de una variable para un período determinado, y no es aceptable que en ejecuciones distintas del modelo de pronóstico se obtengan resultados diferentes. Por ejemplo, el algoritmo genético puede pronosticar en una ejecución que el valor de la variable para el período *x* es 100, pero en otra ejecución pronosticará que el valor de la variable para el mismo período *x* es 300 ó cualquier otro valor distinto.

En segundo lugar, para la evaluación del algoritmo genético (ver sección 2.6) se necesita conocer una función que determine la puntuación (fitness) de cada individuo de la población, ésta es la función de evaluación ó función de ajuste del modulo evolutivo (ver sección 2.2).

En el caso de los pronósticos para la función de evaluación puede utilizarse cualquiera de los métodos de medición de error de pronóstico vistos en la sección 1.4:

- Desviación Absoluta de la Media (DAM)
- Error Medio Cuadrado (EMC)
- Porcentaje de Error Medio Absoluto (PEMA)

- Porcentaje Medio de Error (PME)

Una característica común en estos cuatro métodos es que miden la desviación del valor pronosticado con respecto a un valor real conocido. Algunos miden el error en términos absolutos y otros como un porcentaje, pero todos necesitan conocer valores reales, que se obtienen a través de datos históricos.

La función de evaluación buscará en este caso minimizar el error del pronóstico. Luego de realizar la evaluación, el método de pronóstico debe ser capaz de proyectar valores futuros, para los cuales no se conocen los valores reales. Esto presenta un problema si se utiliza el algoritmo genético como un método de pronóstico directo, pues éste siempre necesitará los datos reales para la función de evaluación.

Por tanto, debe buscarse una representación distinta del método de pronóstico que utilice el algoritmo genético como un medio indirecto para su construcción y que permita pronosticar valores futuros sin necesidad de conocer los valores reales.

3.2. Diseño preliminar del algoritmo genético

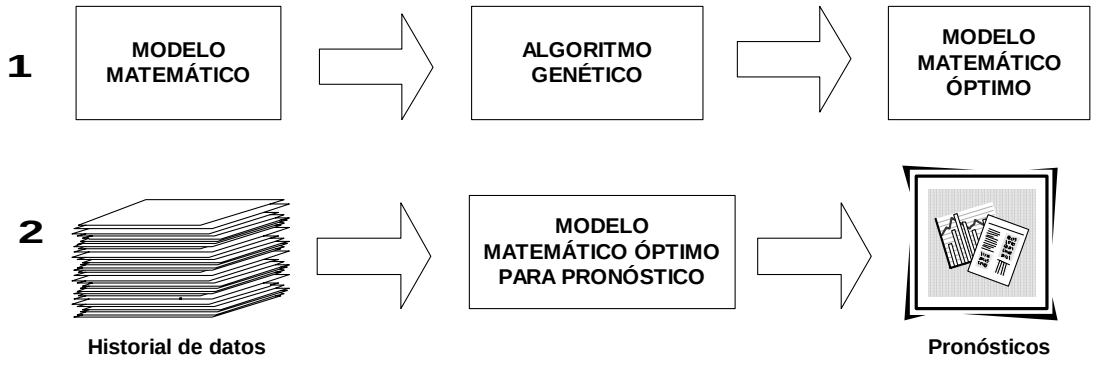
“Los algoritmos genéticos han sido específicamente desarrollados para abordar problemas de optimización de funciones (funciones objetivo).” [5] Por lo tanto, un método de pronóstico que utilice una función matemática como modelo de pronóstico puede ser abordado perfectamente por un algoritmo genético.

Se utilizará entonces un modelo de pronóstico matemático que será evaluado por el algoritmo genético para obtener un modelo de pronóstico óptimo (menor error) que servirá luego para pronosticar valores futuros de la variable en estudio. Así, el algoritmo genético se usa en forma indirecta para pronosticar, evitando los problemas vistos en la sección anterior.

El proceso de pronóstico consistirá entonces de dos pasos, como se muestra en la figura 7:

1. Utilizar el algoritmo genético para construir un modelo matemático óptimo.
2. Utilizar el modelo matemático óptimo para pronosticar.

Figura 7. Diseño preliminar del algoritmo genético



Se debe definir primero el modelo matemático y luego el algoritmo genético que lo evaluará.

3.3. Modelo matemático

Se pretende encontrar un modelo matemático que permita pronosticar valores futuros de una variable, los cuales dependen de un período determinado en el tiempo.

Dentro de los métodos cuantitativos de pronóstico se encuentran los de regresión lineal simple que buscan encontrar una relación entre dos variables, una variable dependiente y y una independiente x, que en su forma más simple esta dada por una ecuación de tipo:

$$y = a + bx$$

Esta ecuación representa la relación lineal que existe entre x y y. En el caso de los pronósticos x es el período en el tiempo y y el valor de la variable en estudio para ese período.

Por ejemplo, si se estudian las ventas mensuales de un producto, x representaría el número de mes y y las ventas correspondientes a ese mes. Así, para encontrar el pronóstico de ventas para un mes determinado, basta con sustituir x por el número de mes y calcular el valor de y.

El modelo de regresión lineal simple tiene la limitante de tener solo dos términos del lado derecho de la ecuación, una constante (a) y un término de primer grado (bx) por lo tanto se ajusta sólo a datos que tengan cierto comportamiento lineal, sin mucha variación.

Existen otras relaciones no lineales que pueden hallarse mediante correlación, basándose en la ecuación básica $y = a + bx$. Así, pueden encontrarse relaciones exponenciales, logarítmicas ó inversas. Pero todas tienen la limitante de poder tratar sólo determinado comportamiento en los datos, sólo exponencial, sólo logarítmico ó solo inverso. Además, en determinadas variables de estudio es difícil encontrar uno de estos comportamientos, por ejemplo, las ventas generalmente no tienen comportamientos exponenciales o logarítmicos.

Para adaptarse a las variaciones en los datos, resulta útil que el modelo tenga varios términos, es decir un polinomio. Como modelo matemático se utilizará un **polinomio de grado *n*** de la forma:

$$y = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

Donde a_1, a_2,…, a_n son los factores numéricos que acompañan a cada término y a_0 es una constante.

Entre mayor sea el grado del polinomio, mayor será la variabilidad de los datos que se pueda modelar.

Si los datos se comportan más o menos uniformemente, por ejemplo, son lineales, entonces algunos factores numéricos tendrán valor de 0.

Así, haciendo los factores a_2, a_3,…, a_n iguales a 0, nos queda un modelo igual al de regresión lineal simple:

$$y = a_0 + a_1 x$$

Si los datos presentan un comportamiento constante, entonces sólo el factor a_0 será distinto a 0.

$$y = a_0$$

Ahora que se ha definido el modelo matemático a utilizar, se diseñará un algoritmo genético que lo evalúe y encuentre los factores numéricos que construyan un polinomio que se ajuste de la mejor manera a los datos en estudio.

3.4. Algoritmo genético

El algoritmo genético debe evaluar cada uno de los factores numéricos que acompañan a los términos del polinomio, de manera que éstos construyan un modelo óptimo que represente correctamente a los datos.

Primero el algoritmo genético recibirá como entrada un conjunto de datos históricos de la variable a pronosticar Y_1, Y_2,…, Y_t, correspondientes a los periodos de tiempo 1, 2,…, t.

Luego buscará los factores numéricos del polinomio a utilizar, A_0, A_1,…, A_n, de forma que éste presente el mínimo error, es decir, la diferencia entre los datos obtenidos con el polinomio $\hat{Y}_1$, $\hat{Y}_2$,…, $\hat{Y}_t$ y los datos reales Y_1, Y_2,…, Y_t sea mínima.

El proceso que seguirá el algoritmo genético para encontrar los factores numéricos del modelo matemático se ilustra en la figura 8 y es el siguiente:

1. El algoritmo genético generará un grupo de individuos, donde cada individuo será un conjunto de factores numéricos aleatorios.

   $a_0, a_1, a_2, \ldots, a_n$

2. Para cada individuo (conjunto de factores numéricos) generado, se obtendrá un conjunto de valores calculados de la siguiente forma:

   $\hat{Y}_1 = a_0 + a_1(1) + a_2(1)^2 + \ldots + a_n(1)^n$

   $\hat{Y}_2 = a_0 + a_1(2) + a_2(2)^2 + \ldots + a_n(2)^n$

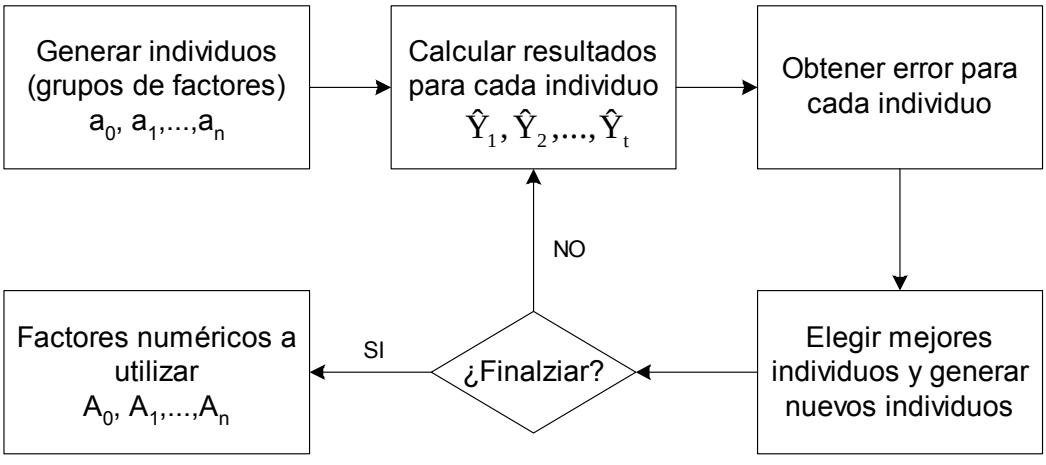
   ...

   $\hat{Y}_t = a_0 + a_1(t) + a_2(t)^2 + \ldots + a_n(t)^n$

3. Se calculará el error comparando los resultados obtenidos contra los datos reales.
4. Se elegirán los mejores individuos, y se generarán nuevos individuos mediante operaciones genéticas.
5. Finalmente se obtendrá el mejor conjunto de factores numéricos

   $A_0, A_1, A_2, \ldots, A_n$

Figura 8. Proceso para encontrar factores numéricos del modelo matemático



De manera más formal, para dar solución a un problema utilizando algoritmos genéticos se deben seguir los siguientes pasos: [14]

- Definir la estructura de los cromosomas o individuos y la representación de los mismos (cada cromosoma es una posible solución potencial para nuestro problema). La estructura del cromosoma depende también del tipo de representación binaria o real de los genes.
- Determinar el criterio de evaluación o adaptación de los individuos. La función de evaluación juega un papel importante en la clasificación potencial de las soluciones en términos de sus características; es el criterio de optimización y evaluación de la calidad de los individuos.
- Determinar los operadores genéticos a utilizar y su forma de aplicación.
- Definir el criterio de parada del algoritmo.
- Definir un criterio de reemplazo entre generaciones consecutivas y el tamaño promedio de las poblaciones.

3.4.1. Estructura de los individuos y su representación

Cada individuo estará compuesto por un conjunto de factores numéricos, uno por cada término en el polinomio. Así, cada individuo tendrá n+1 factores numéricos, pues el primer factor tiene índice 0, donde n es el grado del polinomio a utilizar.

Para un mejor ajuste del polinomio al comportamiento de los datos, se utilizarán números reales como factores numéricos.

El valor de los factores numéricos puede considerarse como cualquier número real, es decir cualquier valor entre menos infinito y más infinito ($-\infty$, $+\infty$). Sin embargo esto no es conveniente debido a que tener un rango muy amplio de posibles valores para cada factor haría lenta e incluso imposible la convergencia del algoritmo genético en algún valor.

Se establecerá entonces un rango fijo para la ejecución del algoritmo genético, denotado por (*-m*, *+m*), dentro del cual se podrán tomar valores reales al azar para cada factor numérico.

La figura 9 ilustra la representación de un individuo dentro del algoritmo genético.

Figura 9. Representación de los individuos

a_0 = Valor aleatorio entre [*-m* , *+m*]	a_1 = Valor aleatorio entre [*-m* , *+m*]	a_2 = Valor aleatorio entre [*-m* , *+m*]	**...**	a_n = Valor aleatorio entre [*-m* , *+m*]

3.4.2. Función de evaluación

La función de evaluación se encarga de calificar a los individuos asignando un puntaje a cada uno, éste punteo suele denominarse *fitness* y representa el grado en que el individuo se aproxima a la solución óptima.

En el caso de los pronósticos, interesa que el valor pronosticado sea lo más cercano al valor real. Esto se mide mediante el error del pronóstico. El error de un pronóstico esta dado por la diferencia entre el valor pronosticado y el valor real.

$$e_t = Y_t - \hat{Y}_t$$

En la sección 1.4 se vieron cuatro métodos para medir el error de un pronóstico:

- Desviación absoluta de la media (DAM): mide el error en las mismas unidades de los datos. Sólo expresa la dimensión del error, no su dirección. Será mejor aquel modelo con el menor DAM.
- Error medio cuadrado (EMC): penaliza los errores grandes pues eleva su valor al cuadrado. Será mejor aquel modelo con el menor EMC.
- Porcentaje de error medio absoluto (PEMA): mide el error en términos relativos, es decir como un porcentaje del valor real. Será mejor aquel modelo con un menor PEMA.
- Porcentaje medio de error (PME): mide el error como un porcentaje del valor real, y además indica el sesgo del pronóstico, es decir si está sobreestimando o subestimando los valores reales, expresando así la dirección del error. Éste es el mejor método para comparar distintos modelos de pronóstico [1]. En este caso, como pueden haber valores negativos, será mejor aquel modelo con un PME cercano a 0.

Cada método mide el error desde diferente enfoque, pero puede utilizarse cualquiera para evaluar un modelo de pronóstico.

La experimentación con el algoritmo genético determinará cuál de los métodos de medición de error es mejor como función objetivo. El PME es un buen método en la comparación de modelos de pronóstico, ya que mide el error como un porcentaje de los valores reales y además indica la dirección del error.

El signo en el valor del PME es de mucha utilidad al momento de elegir un modelo de pronóstico, pues dependiendo la naturaleza de la variable en estudio convendrá más un pronóstico sobreestimado o subestimado.

Por ejemplo, si se estudia la cantidad a producir de algún producto, convendrá más tener un pronóstico sobreestimado para mantener cierto nivel de seguridad en el inventario. Pero si se estudian las ganancias esperadas al final del mes, será mejor tener un pronóstico subestimado que permita estar preparados para una situación con pocas ganancias.

El método PEMA también es útil y puede utilizarse en lugar del PME, sin embargo la utilidad de emplear números con signo se analizará de mejor manera al momento de desarrollar el algoritmo genético.

3.4.3. Operadores genéticos

3.4.3.1 *Crossover*

Se emplearán diversos tipos de *crossover* para determinar mediante experimentación cuál es más adecuado. Se utilizarán los siguientes tipos:

- *Crossover* de un punto
- *Crossover* de dos puntos
- *Crossover* uniforme
- *Crossover* de dos puntos asimétrico

Para una mejor definición de cada uno de ellos ver la sección 2.8.1.

Cada tipo de *crossover* se implementará independientemente, realizando una ejecución del algoritmo genético con cada tipo de *crossover* y comparando al final los resultados obtenidos. Será mejor aquel tipo de *crossover* con el cual el algoritmo genético converja más rápido y genere los mejores individuos.

3.4.3.2 Mutación

La mutación consiste en elegir un individuo al azar entre la población y alterar uno de sus genes.

Para el algoritmo genético en estudio esto se hará de la siguiente forma:

- Seleccionar al azar un individuo (conjunto de factores numéricos) *i* de la población
- Seleccionar al azar un factor numérico a_j del individuo, y sustituirlo por un número al azar entre $-m$ y $+m$, que es el rango de valores posibles para los factores numéricos.

El *crossover* será el operador principal para generar nuevos individuos, y la mutación servirá para mantener la diversidad de la población, evitando que prevalezcan sólo ciertos individuos de la población.

Primero se aplicará *crossover* para generar la nueva población y luego se realizarán mutaciones a algunos individuos de la nueva población elegidos aleatoriamente. Las mutaciones serán mínimas.

3.4.4. Criterio de parada

Existen dos criterios de parada posibles:

- Mediante un número determinado de ejecuciones del algoritmo genético
- Cuando se alcanza un porcentaje de error determinado

En el primer caso, el algoritmo se detiene cuando se alcanza cierto número de ejecuciones, el cual se establece al inicio, por ejemplo, luego de mil ejecuciones.

En el segundo caso, el algoritmo se detiene cuando se logra obtener un individuo con cierto porcentaje de error, por ejemplo, cuando hay un individuo en la población con un error menor o igual a uno por ciento.

Se utilizarán ambos criterios inicialmente, hasta ver mediante experimentación cual es el más adecuado. Por ejemplo, si se observa que luego de mil ejecuciones los individuos están muy lejos de lo óptimo, entonces se incrementará el número de ejecuciones. Así mismo, si se ve que un error del uno por ciento no da individuos con buenas características entonces se reducirá el nivel de error permitido, y si al contrario se ve que transcurrido mucho tiempo ningún individuo llega al nivel de error permitido, entonces se incrementará. En ambos casos, es necesaria la experimentación previa a poder dar valores concretos.

3.4.5. Criterio de reemplazo y tamaño promedio de población

El criterio de reemplazo y tamaño promedio de la población más adecuados también deben establecerse mediante experimentación.

Se iniciará con un tamaño *N* de la población, y *r* individuos para reemplazo. Los individuos que pasarán a la siguiente generación se seleccionarán mediante elitismo (ver sección 2.6.1), es decir, los mejores individuos (con menor error) pasarán a la siguiente generación.

Un criterio alternativo para obtener la nueva población sería seleccionar un grupo de los mejores individuos y cruzarlos para obtener una descendencia que formará la nueva población, esta es una selección generacional (ver sección 2.6.1).

Un tercer criterio a utilizar para seleccionar a los individuos de la nueva población será el hacinamiento determinista (*deterministic crowding*) visto en la sección 2.6.1.1.

Puede utilizarse adicionalmente cualquiera de los métodos de selección vistos en la sección 2.6.1. La elección de uno u otro método dependerá de los resultados que se obtengan en la ejecución del algoritmo genético.

Para el caso de la selección elitista el proceso consiste en copiar cierto número de los mejores individuos a la siguiente generación, el resto de individuos se genera mediante reproducción entre los individuos que han sido copiados.

El tamaño de la población será constante, de *N* individuos. En cada ejecución se elegirán los *r* individuos más aptos y se reemplazarán *N*-*r* individuos mediante reproducción entre los individuos seleccionados. La reproducción se realizará aplicando las operaciones genéticas de *crossover* y mutación que se detallaron en la sección 3.4.3.

Si la reproducción es generacional, se eligen los *r* mejores individuos de la población actual y se reproducen para generar *N* nuevos individuos que formarán la nueva generación. No se copia ningún individuo de la población actual a la siguiente.

3.5. Diseño final del algoritmo genético

Luego de definir todos los aspectos del algoritmo genético, se resumirá su diseño en el siguiente flujo:

1. Leer conjunto de datos históricos.

   $Y_1, Y_2,\ldots, Y_n$

2. Definir parámetros del algoritmo genético:
   - Grado del polinomio a utilizar *n*
   - Rango de valores para los factores numéricos *–m*, *+m*
   - Tipo de función de evaluación a utilizar, que puede ser cualquiera de los métodos de medición de error de pronóstico
   - Tipo de *crossover*
   - Número de mutaciones en cada generación *m*
   - Tamaño de la población *N*
   - Tipo de selección a utilizar, elitista o generacional
   - Número de individuos mejores a seleccionar en cada generación *r*
   - Condición de finalización: Número de ejecuciones ITERACIONES o porcentaje medio de error mínimo deseado ERR_MIN
3. Generar población inicial
   - Mientras no se hayan generado *N* individuos
     - Generar *n*+1 factores numéricos al azar $a_0, a_1,\ldots, a_n$. Cada uno es un valor aleatorio entre *–m* y *+m*
4. Calcular el valor de la función objetivo para cada individuo
   - Calcular los valores pronosticados con el individuo *i*

     $\hat{Y}_{1i} = a_{0i} + a_{1i}\,(1) + a_{2i}\,(1)^2 + \ldots + a_{ni}\,(1)^n$

     $\hat{Y}_{2i} = a_{0i} + a_{1i}\,(2) + a_{2i}\,(2)^2 + \ldots + a_{ni}\,(2)^n$

     ...

     $\hat{Y}_{ti} = a_{0i} + a_{1i}\,(t) + a_{2i}\,(t)^2 + \ldots + a_{ni}\,(t)^n$

- Calcular el error del pronóstico mediante el método seleccionado

$$DAM = \frac{\sum_{t=1}^{n} \left| Y_t - \hat{Y}_t \right|}{n}$$

$$EMC = \frac{\sum_{t=1}^{n} \left(Y_t - \hat{Y}_t \right)^2}{n}$$

$$PME = \frac{\sum_{t=1}^{n} \frac{\left(Y_t - \hat{Y}_t \right)}{Y_t}}{n}$$

$$PEMA = \frac{\sum_{t=1}^{n} \frac{\left| Y_t - \hat{Y}_t \right|}{Y_t}}{n}$$

- Repetir hasta terminar los *N* individuos de la población

5. Tomar los *r* mejores individuos de la población actual
6. Generar los individuos para la siguiente generación (nueva población)
   - Si la selección es elitista
     - Copiar los *r* mejores individuos de la población actual a la nueva población
     - Generar *N-r* individuos mediante crossover
       - Tomar dos de los *r* mejores individuos al azar
       - Aplicar *crossover*
   - Si la selección es generacional
     - Generar *N* individuos mediante *crossover*
       - Tomar dos de los *r* mejores individuos al azar
       - Aplicar *crossover*
   - Si la selección es mediante hacinamiento determinista
     - Seleccionar parejas de individuos y generar dos hijos por cada pareja mediante *crossover*

        - Reemplazar a los individuos de la población original si los hijos son mejores
    - Aplicar mutación a *m* individuos de la nueva población
        - Tomar un individuo al azar entre todos los individuos de la nueva población
        - Cambiar un factor numérico al azar entre los *n*+1 factores
7. Verificar si se ha alcanzado la condición de finalización:
    - número de ejecuciones >= ITERACIONES ó ERROR del mejor individuo <= ERR_MIN
        - SI: Pasar al punto 8
        - NO: Con la nueva población repetir los pasos 4 al 6
8. Elegir el mejor individuo de la población para construir el polinomio que servirá como modelo matemático.
9. Fin

4. DESARROLLO DE LA APLICACIÓN

4.1. Lenguaje de programación

El lenguaje de programación que se empleó para el desarrollo de la aplicación fue Java. Como editor se utilizó Borland Java Builder 9.

Se eligió Java debido a que la mayor parte de documentación y ejemplos sobre algoritmos genéticos encontrados estaban en este lenguaje, lo que permitió reutilizar parte del código encontrado.

Pudo utilizarse el lenguaje C, debido a que también existen varios algoritmos genéticos en este lenguaje, pero Java tiene la ventaja de ser un lenguaje gráfico lo que provee una mejor interfaz para la evaluación de los resultados.

4.2. Clases principales de la aplicación

4.2.1. Parámetros

Los parámetros para la ejecución del algoritmo genético se almacenan en un archivo XML, el cual puede modificarse mientras se ejecuta la aplicación. Para leer y escribir al archivo XML se ha empleado la librería JDOM, incluida en JBuilder. Esta librería permite crear y leer una estructura XML de manera flexible. Se puede descargar la última versión de esta librería en [15].

La estructura del archivo XML utilizado para almacenar los parámetros es la siguiente:

```
<parametros>
    <parametro>
        <nombre> </nombre>
        <valor></valor>
    </parametro>
</parametros>
```

Los parámetros que utiliza la aplicación son:

- Grado: grado del polinomio que se utilizará como modelo matemático para pronosticar. Este determina la longitud de los individuos de la población para el algoritmo genético.
- Factor-min y Factor-max: límites para los factores numéricos del polinomio. Estos se utilizan al momento de generar la población inicial compuesta por números aleatorios entre este rango.
- Población: Tamaño de la población (*N*). Número de individuos que tiene la población.
- Tipo-crossover: tipo de crossover a utilizar (1=De un punto, 2=De dos puntos, 3=Uniforme, 4=De dos puntos asimétrico)
- Patrón: patrón binario a utilizar para el crossover uniforme.
- Mutaciones: número de mutaciones a realizar por generación. Es el número de individuos que serán mutados cada vez que se genere una nueva población.
- Mutar mejor individuo: indica si debe mutarse el mejor individuo de cada generación. Si no se muta el mejor individuo, se garantiza su existencia en las siguientes generaciones hasta que se encuentre uno mejor.

- Tipo de función de evaluación: tipo de función objetivo a utilizar, puede ser cualquiera de los métodos de medición de error en el pronóstico.
- Divisor: divisor a utilizar cuando se calcula la función de evaluación para evitar valores muy grandes que puedan causar un error de desbordamiento de tipo de dato durante la ejecución del programa.
- Tipo de selección: indica el tipo de selección que se utilizará para generar la nueva población. Puede ser elitista, generacional o hacinamiento determinista.
- Mejores: indica el número de individuos que se utilizará para generar la siguiente población. Si el proceso de selección es elitista, entonces éste número de individuos se copiará a la siguiente generación y el resto se generará mediante reproducción genética entre los que fueron copiados. Cuando el proceso de selección es generacional, se elegirá éste número de mejores individuos de la generación actual y se reproducirán para crear una nueva generación formada sólo por su descendencia. Este dato no se utiliza si el proceso de selección es por hacinamiento determinista.
- Condición-fin: condición de terminación del algoritmo genético (1=Número de ejecuciones, 2=Porcentaje de error aceptado).
- Valor-fin: valor que indica la finalización de la ejecución del algoritmo genético. Dependiendo la condición de finalización puede ser un número o un porcentaje.

Los parámetros se leen y almacenan en constantes para evitar ir a leer el archivo cada vez que se necesite acceder a su valor.

La función que lee el valor de un parámetro se muestra en la figura 10.

Esta función, recibe el nombre del parámetro a consultar y retorna su valor en una cadena. La función recorre todos los nodos del archivo XML de parámetros hasta encontrar un nodo cuyo elemento nombre coincida con el nombre indicado.

Figura 10. Función que lee el valor de un parámetro del archivo XML

```
Public String ValorParametro(String strNombre) {
   SAXBuilder builder;
   Document doc;

   Element eRaiz;
   Element eParam;
   Element eNombre;
   Element eValor;
   List lParametros;
   Iterator iParam;
   boolean blnEncontrado;
   String strValor = new String("");

   builder = new SAXBuilder(false);
   doc = builder.build(C_ARCHIVO_PARAMETROS);
   eRaiz = doc.getRootElement();
   lParametros = eRaiz.getChildren("parametro");
   iParam = lParametros.iterator();

   blnEncontrado = false;
   // Recorriendo lista de parámetros
   while (iParam.hasNext() && !blnEncontrado) {
     eParam = (Element)iParam.next();
     eNombre = eParam.getChild("nombre");
     if (eNombre.getText().equals(strNombre)) {
       eValor = eParam.getChild("valor");
       strValor = eValor.getText();
       blnEncontrado = true;
     }
   }
   return strValor;
}
```

4.2.2. Historial de datos

La aplicación utiliza un historial de datos que sirve para evaluar el modelo matemático que encuentra el algoritmo genético. Este historial de datos esta compuesto por una serie de tiempo, es decir un conjunto de valores en el tiempo de la variable en estudio.

Los valores se leen de un archivo que contiene un valor por cada línea. La primera línea contiene el valor de la variable para el primer período, sea mes, día o año, la segunda línea contiene el valor para el segundo período y así sucesivamente.

De manera formal, el historial de datos contiene un conjunto de valores *f(x)* para la variable *x* en estudio. Estos valores le sirven al algoritmo genético para encontrar una aproximación a la función *f*, la que se convertirá en el modelo matemático a utilizar para realizar pronósticos de la variable *x*.

El historial se almacena en un arreglo de una dimensión, en el cual cada posición representa un período de tiempo iniciando en la posición cero.

4.2.3. Población

La población del algoritmo genético esta formada por una matriz de números reales, donde cada fila representa un individuo. La matriz tiene un número de columnas igual al grado del polinomio definido en los parámetros y un número de filas igual al número de individuos, también definido en los parámetros.

La población inicial esta formada por individuos generados aleatoriamente. Cada individuo es un conjunto de factores numéricos generados al azar que varían dentro el rango especificado en los parámetros. El procedimiento de generación de la población inicial se muestra en la figura 11.

Figura 11. Generación de población inicial

```
Public void PoblacionInicial() {
  int i, j;
  for (i=0; i<Constantes.C_TAM_POBLACION; i++) {
    for (j=0; j<Constantes.C_LONG_INDIVIDUO; j++) {
      arrPoblacion[i][j] = GeneraFactor();
    }
  }
  lngGeneracion = 1;
}
```

La función que genera un factor numérico aleatorio primero calcula el rango del intervalo, luego genera un número aleatorio entre cero y el valor del rango, finalmente lo suma al límite inferior del rango para obtener el factor numérico. Por ejemplo, si el rango es [-10, 10] entonces el valor del rango es 20, se genera un número aleatorio entre 0 y 20, digamos 5, y se suma este número al límite inferior que es -10, entonces el factor numérico resultante es -5. Esto se muestra en la figura 12.

Figura 12. Función que genera un factor aleatorio

```
private float GeneraFactor() {
    float fltRango;
    float fltResultado = 0;

    fltRango = Constantes.C_FACTOR_MAX  - Constantes.C_FACTOR_MIN;
    fltResultado = Constantes.C_FACTOR_MIN + (fltRango *
                                          new Double(Math.random()).floatValue());
    return fltResultado;
}
```

4.2.4. Proceso de selección

Si el proceso de selección a utilizar es elitista o generacional es necesario obtener los individuos con mejor punteo, esto se logra ordenando los individuos de menor a mayor en base a su puntuación. El ordenamiento es ascendente debido a que la función de evaluación mide el error de los datos pronosticados con respecto a los datos reales, entonces serán mejores aquellos individuos con menor punteo, es decir con menor error.

Para el caso del hacinamiento determinista no es necesario realizar un ordenamiento pues las parejas de individuos se eligen aleatoriamente.

Para ordenar a los individuos de la población, se utiliza otra matriz con dos columnas y una fila por cada individuo. La primera columna almacena la posición del individuo en la matriz de población y la segunda columna el punteo correspondiente a ese individuo, como se muestra en la tabla VII.

Tabla VII. Estructura del arreglo de punteos

Posición del individuo en la población	*Punteo*
1	P_1
2	P_2
…	…
N	P_N

La matriz de punteos es ordenada en base a los valores de la columna de punteo. Una vez ordenada la matriz de punteos, se puede saber el orden de los individuos leyendo la columna de posiciones.

La función de ordenamiento se extrajo del paquete *playground* [16], y fue modificada para manejar un arreglo de dos dimensiones, la función se muestra en la figura 13. Esta función recibe un arreglo de dos dimensiones (posición, punteo) y lo ordena en base a la columna de punteo. La función recibe como parámetro si el ordenamiento debe realizarse en forma ascendente o descendente, para el algoritmo genético desarrollado se utilizó el ordenamiento ascendente debido a que son mejores aquellos individuos con menor punteo.

Figura 13. Función de ordenamiento

```
public double[][] Ordenar(double arrDatos[][], boolean descendente) {
  int inc, comp;
  double v; // Valor a comparar
  double p; // Posición a comparar
  if (descendente)
    comp = -1;
  else
    comp = 1;

  for (inc = 1; inc <= arrDatos.length / 9; inc = 3 * inc + 1);

  for (; inc > 0; inc /= 3) {
    for (int i = inc + 1; i <= arrDatos.length; i += inc) {
      v = arrDatos[i - 1][1];
      p = arrDatos[i - 1][0];
      int j = i;
      while ( (j > inc) && (Comparar(arrDatos[j - inc - 1][1], v) == comp)) {
        // Copiando posición
        arrDatos[j - 1][0] = arrDatos[j - inc - 1][0];
        // Copiando punteo
        arrDatos[j - 1][1] = arrDatos[j - inc - 1][1];
        j -= inc;
      }
      arrDatos[j - 1][1] = v;
      arrDatos[j - 1][0] = p;
    }
  }
  return arrDatos;
}
```

La función de ordenamiento llama a una función que compara dos valores reales, y devuelve -1 si el primer valor es menor al segundo, 1 si el primer valor es mayor al segundo y 0 si los dos valores son iguales. Ésta función se muestra en la figura 14.

Figura 14. Función de comparación

```
public static int Comparar(double x1, double x2) {
    int flag;

    if (x1 < x2)
      flag = -1;
    else if (x1 > x2)
      flag = 1;
    else
      flag = 0;

    return flag;
  }
```

Luego de haber generado la primera población, es necesario crear nuevas generaciones mediante un proceso de selección y reproducción entre los individuos de la población actual.

Si la selección es elitista, entonces se copian los mejores individuos de la población actual a la siguiente generación, esto se hace obteniendo primero las posiciones de los mejores individuos del arreglo de punteos y luego copiando esas posiciones a la nueva población. Luego, se genera el resto de individuos mediante *crossover*. Para ello primero se eligen dos individuos, padre y madre, entre los mejores de la población actual. El ciclo para el *crossover* inicia a partir del número de individuos copiados anteriormente, este proceso se muestra en la figura 15.

Figura 15. Selección de individuos elitista

```
tamCopiar = Constantes.C_TAM_POBLACION - Constantes.C_NUM_REEMPLAZO;
for (i=0; i<tamCopiar; i++) {
  // Primero se obtiene la posición y luego se copia
  pos = new Double(arrPunteo[i][0]).intValue();
  arrNuevaPob[i] = arrPoblacion[pos];
}
i = tamCopiar;
while (i<Constantes.C_TAM_POBLACION) {
  // Buscar posiciones de individuos padre y madre
  posPadre = new Double(Math.floor(Math.random() * numMejores)).intValue();
  do {
    posMadre = new Double(Math.floor(Math.random() * numMejores)).intValue();
  } while (posPadre == posMadre);
  // Obtener del arreglo de punteos las posiciones en la población
  posPadre = new Double(arrPunteo[posPadre][0]).intValue();
  posMadre = new Double(arrPunteo[posMadre][0]).intValue();
```

Si el proceso de selección es generacional, no se copia ningún individuo de la población actual a la siguiente generación, sino que se eligen los mejores individuos de la población actual y su descendencia formará la totalidad de la siguiente generación. Esto se muestra en la figura 16.

Figura 16. Selección de individuos generacional

```
i=0;
while (i<Constantes.C_TAM_POBLACION) {
  // Buscar posiciones de individuos padre y madre
  posPadre = new Double(Math.floor(Math.random() * numMejores)).intValue();
  do {
    posMadre = new Double(Math.floor(Math.random() * numMejores)).intValue();
  } while (posPadre == posMadre);
  // Obtener del arreglo de punteos las posiciones en la población
  posPadre = new Double(arrPunteo[posPadre][0]).intValue();
  posMadre = new Double(arrPunteo[posMadre][0]).intValue();
```

La diferencia entre la selección elitista y la generacional, es que la primera inicia el ciclo de *crossover* a partir del número de individuos que fueron copiados, mientras que la segunda inicia el ciclo desde cero porque todos los individuos serán generados mediante *crossover*.

Cuando la selección es mediante hacinamiento determinista se toma como base toda la población actual y se seleccionan aleatoriamente dos individuos de ésta para combinarlos genéticamente. Como puede observarse en la figura 17 se incluye el tamaño total de la población para obtener las posiciones de los padres y no solo cierto número de los mejores individuos como se hace en la selección elitista y generacional, además no se toma en cuenta la posición que tengan los individuos en el arreglo de punteos.

Figura 17. Selección de individuos con hacinamiento determinista

```
While (i < Constantes.C_TAM_POBLACION) {
  // Buscar posiciones de individuos padre y madre
  posPadre1 = new Double(Math.floor(Math.random() *
                                    Constantes.C_TAM_POBLACION)).intValue();
  do {
    posPadre2 = new Double(Math.floor(Math.random() *
                                     Constantes.C_TAM_POBLACION)).intValue();
  }
  while (posPadre1 == posPadre2);
  …
```

Una vez se tiene a los individuos padres se procede a generar dos hijos mediante *crossover*. Si el *crossover* es de un punto primero se genera aleatoriamente un punto de intercambio k1 y luego se llama a la función que realiza el intercambio genético, indicando en el último parámetro la manera en que se realizará el intercambio. Esto se muestra en la figura 18.

Figura 18. Generación de hijos mediante *crossover* de un punto

```
// Calcular punto de intercambio k1
do {
  k1 = (int) Math.floor(Constantes.C_LONG_INDIVIDUO * Math.random());
} while (k1 == 0);
// Generar primer hijo padre-madre
arrNuevaPob[i] = op.crossover1p(arrPoblacion[posPadre],arrPoblacion[posMadre],k1,1);
// Generar segundo hijo madre-padre
arrNuevaPob[i] = op.crossover1p(arrPoblacion[posPadre],arrPoblacion[posMadre],k1,2);
```

Cuando el *crossover* es de dos puntos, se generan dos puntos de intercambio, k1 y k2, asegurando que el segundo punto sea mayor al primero. Luego se llama a la función que realiza el intercambio genético, indicando en el último parámetro la manera en que se realizará el intercambio. Esto se muestra en la figura 19.

Figura 19. Generación de hijos mediante *crossover* de dos puntos

```
// Calculando primer punto de intercambio k1
do {
  k1 = (int) Math.floor(Constantes.C_LONG_INDIVIDUO * Math.random() * 0.66);
} while (k1 == 0);
// Calculando segundo punto de intercambio k2
do {
  k2 = (int) Math.floor(Constantes.C_LONG_INDIVIDUO * Math.random());
} while (k1 >= k2);

// Generar primer hijo padre-madre-padre
arrNuevaPob[i] = op.crossover2p(arrPoblacion[posPadre],arrPoblacion[posMadre],k1,k2,1);
// Generar segundo hijo madre-padre-madre
arrNuevaPob[i] = op.crossover2p(arrPoblacion[posPadre],arrPoblacion[posMadre],k1,k2,2);
```

Si el *crossover* es uniforme sólo se llama a la función que realiza el intercambio enviando como parámetro el patrón binario a utilizar, además se indica en el último parámetro la forma en que se realizará el intercambio. Esto se muestra en la figura 20.

Figura 20. Generación de hijos mediante *crossover* uniforme

```
// Generar primer hijo 1=padre, 0=madre
arrNuevaPob[i] = op.crossoverUniforme(arrPoblacion[posPadre], arrPoblacion[posMadre],
                                      Constantes.C_PATRON, 1);
// Generar segundo hijo 1=madre, 0=padre
arrNuevaPob[i] = op.crossoverUniforme(arrPoblacion[posPadre], arrPoblacion[posMadre],
                                      Constantes.C_PATRON, 2);
```

Cuando el *crossover* es de dos puntos asimétrico se generan dos puntos de intercambio en el padre, k1 y k2, asegurando que el segundo punto sea mayor al primero y un punto para iniciar la copia en la madre, k3. Luego se llama a la función que realiza el intercambio indicando en el último parámetro la forma en que se hará dicho intercambio. Esto se muestra en la figura 21.

Figura 21. Generación de hijos mediante *crossover* de dos puntos asimétrico

```
// Calculando punto inicial en padre k1
k1 = (int) Math.floor(Constantes.C_LONG_INDIVIDUO * Math.random() * 0.66);
// Calculando punto final en padre k2
do {
  k2 = (int) Math.floor(Constantes.C_LONG_INDIVIDUO * Math.random());
} while (k1 >= k2);
// Calculando punto en madre
k3 = (int) Math.floor(Constantes.C_LONG_INDIVIDUO * Math.random());

// Generar primer hijo
arrNuevaPob[i] = op.atc(arrPoblacion[posPadre], arrPoblacion[posMadre], k1, k2, k3, 1);
// Generar segundo hijo
arrNuevaPob[i] = op.atc(arrPoblacion[posPadre], arrPoblacion[posMadre], k1, k2, k3, 2);
```

Cuando se realiza hacinamiento determinista los individuos hijos no pasan a formar parte de la nueva población directamente como sucede en la selección generacional y elitista, sino se realiza un proceso adicional de evaluación de los hijos antes de que reemplacen a los padres de la población original.

Luego de que cada pareja de padres ha generado una pareja de hijos, se compara cada hijo con el padre con que es más similar de acuerdo a la distancia de Hamming. La distancia de Hamming se obtiene al determinar el número de posiciones que son distintas entre el arreglo padre y el arreglo hijo. Si el punteo de un hijo es mejor que el de un padre entonces se reemplaza al padre, sino entonces el padre permanece en la nueva población (ver sección 2.6.1.1).

El procedimiento inicia obteniendo los punteos de los padres y de los hijos, luego se verifica si los padres son iguales en cuyo caso se compara un padre con un hijo y se reemplaza el padre si el hijo tiene mejor punteo, como se muestra en la figura 22.

Cuando los padres son distintos se verifica que hijo se debe comparar con que padre en base a la distancia de Hamming (posiciones distintas) que existe entre los arreglos, como se observa en la figura 23.

Figura 22. Reemplazo de individuos en hacinamiento determinista cuando los padres iguales

```
// Calculo de punteos para comparaciones
punteoP1 = eval.Punteo(arrPadre1);
punteoP2 = eval.Punteo(arrPadre2);
punteoH1 = eval.Punteo(arrHijo1);
punteoH2 = eval.Punteo(arrHijo2);

// Si los padres son iguales, tomar el hijo con mejor punteo
if (arrPoblacion[posPadre1] == arrPoblacion[posPadre2]) {
  if (punteoH1 < punteoP1)
    arrPoblacion[posPadre1] = arrHijo1;
  if (punteoH2 < punteoP1)
    arrPoblacion[posPadre2] = arrHijo2;
}
```

Figura 23. Obtención de parejas padre-hijo a comparar en hacinamiento determinista

```
  // Obtener padre más similar al hijo1 para comparar
  if (DistanciaHamming(arrPadre1,arrHijo1) < DistanciaHamming(arrPadre2,arrHijo1))
    padreComparaHijo1 = 1;
  else
    padreComparaHijo1 = 2;
  // Obtener padre más similar al hijo2 para comparar
  if (DistanciaHamming(arrPadre1,arrHijo2) < DistanciaHamming(arrPadre2,arrHijo2))
    padreComparaHijo2 = 1;
  else
    padreComparaHijo2 = 2;
```

Cuando cada hijo es más similar a un padre distinto, es decir cada pareja a comparar padre-hijo es diferente, entonces se reemplaza el padre si el hijo con el que se compara tiene mejor punteo, como se muestra en la figura 24.

Figura 24. Reemplazo de individuos en hacinamiento determinista, parejas padre-hijo diferentes

```
  If (padreComparaHijo1 ¡= padreComparaHijo2) {
    // Los padres a comparar son distintos
    if (padreComparaHijo1 == 1) {
      // Comparar hijo1 con padre1 e hijo2 con padre2
      if (punteoH1 < punteoP1)
        arrPoblacion[posPadre1] = arrHijo1;
      if (punteoH2 < punteoP2)
        arrPoblacion[posPadre2] = arrHijo2;
    }
    else {
      // Comparar hijo1 con padre2 e hijo2 con padre1
      if (punteoH1 < punteoP2)
        arrPoblacion[posPadre2] = arrHijo1;
      if (punteoH2 < punteoP1)
        arrPoblacion[posPadre1] = arrHijo2;
    }
  }
```

Si ambos hijos son más similares al mismo padre, es decir las parejas a comparar padre-hijo son iguales, entonces:

- Si los dos hijos son mejores que el padre entonces se toma al hijo más similar de acuerdo a la distancia de Hamming.
- Si no se reemplaza el padre por el hijo que es mejor, o se mantiene en caso de que ninguno de los hijos sea mejor.

Este proceso de reemplazo, tomando en cuenta los casos cuando ambos hijos son más similares al primer o segundo padre, se muestra en la figura 25.

Figura 25. Reemplazo de individuos en hacinamiento determinista, parejas padre-hijo iguales

```
If (padreComparaHijo1 == padreComparaHijo2) {
  // Ambos hijos son más similares al mismo padre
  if (padreComparaHijo1 == 1) {
    // Comparar ambos hijos con padre1
    if ((punteoH1 < punteoP1) && (punteoH2 < punteoP1)) {
      // Ambos hijos son mejores que el padre1, tomar el hijo más similar
      if (DistanciaHamming(arrPadre1,arrHijo1) < DistanciaHamming(arrPadre1,arrHijo2))
        arrPoblacion[posPadre1] = arrHijo1;
      else
        arrPoblacion[posPadre1] = arrHijo2;
    }
    else {
      // Solo uno o ninguno de los dos hijos es mejor que el padre1
      // Comparar hijos individualmente
      if (punteoH1 < punteoP1)
        arrPoblacion[posPadre1] = arrHijo1;
      else if (punteoH2 < punteoP1)
        arrPoblacion[posPadre1] = arrHijo2;
    }
  }
  else {
    // Comparar ambos hijos con padre2
    if ((punteoH1 < punteoP2) && (punteoH2 < punteoP2)) {
      // Ambos hijos son mejores que el padre 2, tomar hijo más similar
      if (DistanciaHamming(arrPadre2,arrHijo1) < DistanciaHamming(arrPadre2,arrHijo2))
        arrPoblacion[posPadre2] = arrHijo1;
      else
        arrPoblacion[posPadre2] = arrHijo2;
    }
    else {
      // Solo uno o ninguno de los dos hijos es mejor que el padre2
      // Comparar hijos individualmente
      if (punteoH1 < punteoP2)
        arrPoblacion[posPadre2] = arrHijo1;
      else if (punteoH2 < punteoP2)
        arrPoblacion[posPadre2] = arrHijo2;
    }
  }
}
```

Cuando se tiene la nueva población, se realizan mutaciones a cierto número de individuos (definido como parámetro) para introducir cierta diversidad en la población. Al momento de realizar la mutación se verifica si debe mantenerse al mejor individuo de la población actual, en cuyo caso debe evitarse mutar al individuo en la primera posición pues es el mejor de la población actual que ha sido copiado a la nueva población, esto para el caso de la selección generacional y elitista, para el hacinamiento determinista no es aplicable porque el mejor individuo no esta necesariamente en la primera posición. El proceso de mutación se muestra en la figura 26.

Figura 26. Proceso de mutación

```
for (i = 0; i<Constantes.C_NUM_MUTACIONES; i++) {
  // Posición del individuo a mutar
  if (Constantes.C_MUTAR_MEJOR == 1) {
    // Se pueden mutar todos los individuos
    posMutacion = new Double(Math.floor(Math.random()*
                             Constantes.C_TAM_POBLACION)).intValue();
  }
  else if (Constantes.C_MUTAR_MEJOR == 2) {
    // No mutar el mejor individuo (posición 0)
    do {
      posMutacion = new Double(Math.floor(Math.random() *
                               Constantes.C_TAM_POBLACION)).intValue();
    } while (posMutacion == 0);
  }
  arrPoblacion[posMutacion] = op.mutacion(arrPoblacion[posMutacion]);
}
```

Una vez se tiene la nueva población, el proceso se repite hasta que se cumpla la condición de finalización.

4.2.5. Función de evaluación

El algoritmo genético utiliza tres funciones de evaluación: desviación absoluta de la media (DAM), error medio cuadrado (EMC) y porcentaje de error medio absoluto (PEMA).

No se utilizó el porcentaje medio de error (PME) debido a que la función de selección necesita que sean mejores aquellos individuos con menor punteo, como los otros tres métodos, y el PME califica como mejores aquellos individuos con punteo cercano a cero, no necesariamente los de menor punteo.

Por ejemplo, el PME puede devolver los siguientes punteos para cinco individuos: 1, 3, -5, -2 y 6, el mejor individuo es el que tiene punteo 1 porque tiene menor error, sin embargo el proceso de selección elegiría el individuo con punteo -5 porque es el menor punteo, alejando al algoritmo genético de una mejor solución. Con el PEMA no ocurre este problema debido a que trabaja sólo con valores absolutos y por lo tanto devuelve sólo punteos positivos.

La función que calcula el punteo de un individuo, recibe como parámetro un arreglo de números y devuelve su punteo. Primero se verifica el tipo de función configurada como parámetro y luego se aplica como se muestra en la figura 27.

Figura 27. Función para cálculo del punteo de un individuo

```
public double Punteo(double[] arrIndividuo) {
    double dblResultado = 0;
    float Yr;
    double Yc;
    int i, n;

    // DAM = Sumatoria_1_n |Y_real - Y_calculado| / n
    // EMC = Sumatoria_1_n (Y_real - Y_calculado)^2 / n
    // PEMA = Sumatoria_1_n (|Y_real - Y_calculado| / Y_real) / n
    n = Constantes.C_NUM_PERIODOS_HISTORIAL;
    for (i = 0; i<n; i++) {
      Yr = arrHistorial[i];
      Yc = YCalculado(i+1, arrIndividuo);
      if (Constantes.C_TIPO_FUNCION_EVALUACION == 0)
        // DAM
        dblResultado = dblResultado + Math.abs(Yr - Yc) / Constantes.C_DIVISOR;
      else if (Constantes.C_TIPO_FUNCION_EVALUACION == 1)
        // EMC
        dblResultado = dblResultado + Math.pow((Yr - Yc), 2) / Constantes.C_DIVISOR;
      else if (Constantes.C_TIPO_FUNCION_EVALUACION == 2)
        // PEMA
        dblResultado = dblResultado + (Math.abs(Yr - Yc) / Yr) / Constantes.C_DIVISOR;
    }
    dblResultado = dblResultado / n;

    return dblResultado;
  }
```

La función que calcula el punteo toma los valores reales de *Y* del historial y los compara contra los valores calculados de *Y* obtenidos con el polinomio o individuo que recibe como parámetro. Los valores calculados de *Y* se obtienen evaluando el polinomio para cada período de tiempo (1, 2, 3,…). La función que calcula el valor de *Y* se muestra en la figura 28.

Figura 28. Función para calcular el valor de Y

```
private double YCalculado(double X, double[] arrIndividuo) {
  double dblCalculado = 0;
  int i;
  // Y = i0 * X^0 + i1 * X^1 + i2 * X^2 + ...
  for (i=0; i<Constantes.C_LONG_INDIVIDUO; i++) {
    dblCalculado = dblCalculado + arrIndividuo[i] * Math.pow(X, i);
  }
  return dblCalculado;
}
```

Esta función evalúa el individuo o polinomio (arreglo de una dimensión) en el período *X* que recibe como parámetro, de la siguiente manera:

$$Y = Individuo[0] * X^0 + Individuo[1] * X^1 + \ldots + Individuo[n] * X^n$$

La función de evaluación tiene una limitante, debido a que para un polinomio de grado *n* muy grande los valores pueden llegar a desbordar el tipo de dato. Aunque en Java no se interrumpe la ejecución del programa cuando el valor excede la capacidad del tipo de dato, devuelve la cadena “infinity” indicando que el valor es infinito, y esto ya no es útil en el proceso de selección de los mejores individuos.

Para evitar este problema existe un parámetro que permite indicar un divisor que se aplicará a la función de evaluación, así se divide cada diferencia de función de evaluación entre el divisor indicado. Para un divisor específico, la función de evaluación quedaría de la siguiente manera:

```
dblResultado = dblResultado + (Math.abs(Yr - Yc) / Yr) / 1000000;
```

4.2.6. Operadores genéticos

Como operadores genéticos se emplearon crossover de uno, dos puntos, uniforme, dos puntos asimétrico y mutación. Todas las funciones de operadores genéticos reciben un conjunto de parámetros y retornan un arreglo de una dimensión con el individuo resultante de aplicar el operador genético.

La función que realiza el crossover de un punto se muestra en la figura 29. Ésta función recibe los siguientes parámetros:

- arrPadre: arreglo de una dimensión que será el padre del nuevo individuo.
- arrMadre: arreglo de una dimensión que será la madre del nuevo individuo.
- k: punto para realizar el intercambio genético.
- intHijo: número de hijo a generar. Indica el orden en que se mezclarán los padres para generar el nuevo hijo, es decir si será padre-madre o madre-padre.

Figura 29. Función de *crossover* de un punto

```
Public float[] crossover1p(float arrPadre[], float arrMadre[], int k, int intHijo) {
   float arrHijo[] = new float[Constantes.C_LONG_INDIVIDUO];
   int i;
   // Copiando material genético del primer individuo
   for (i = 0; i < k; i++) {
     if (intHijo == 1) // Tomar primero el material genético del padre
       arrHijo[i] = arrPadre[i];
     else if (intHijo == 2) // Tomar primero el material genético de la madre
       arrHijo[i] = arrMadre[i];
   }
   // Copiando material genético del segundo individuo
   for (i = k; i < Constantes.C_LONG_INDIVIDUO; i++) {
     if (intHijo == 1) // Tomar el material genético de la madre
       arrHijo[i] = arrMadre[i];
     else if (intHijo == 2) // Tomar el material genético del padre
       arrHijo[i] = arrPadre[i];
   }
   return arrHijo;
 }
```

La función que realiza el *crossover* de dos puntos se muestra en la figura 30. Ésta función recibe los siguientes parámetros:

- arrPadre: arreglo de una dimensión que será el padre del nuevo individuo.
- arrMadre: arreglo de una dimensión que será la madre del nuevo individuo.
- k1: primer punto para intercambio genético.
- k2: segundo punto para intercambio genético.
- intHijo: número de hijo a generar. Indica el orden en que se mezclarán los padres para generar el nuevo hijo, que puede ser padre-madre-padre o madre-padre-madre.

Figura 30. Función de *crossover* de dos puntos

```
Public float[] crossover2p_pos(float arrPadre[], float arrMadre[], int k1,
                               int k2, int intHijo) {
   float arrHijo[] = new float[Constantes.C_LONG_INDIVIDUO];
   int i;

   if (intHijo == 1) {
     // Copiando material genético del padre
     for (i = 0; i < k1; i++)
       arrHijo[i] = arrPadre[i];
     // Copiando material genético de la madre
     for (i = k1; i < k2; i++)
       arrHijo[i] = arrMadre[i];
     // Copiando material genético del padre
     for (i = k2; i < Constantes.C_LONG_INDIVIDUO; i++)
       arrHijo[i] = arrPadre[i];
   }
   else if (intHijo == 2) {
     // Copiando material genético de la madre
     for (i = 0; i < k1; i++)
       arrHijo[i] = arrMadre[i];
     // Copiando material genético del padre
     for (i = k1; i < k2; i++)
       arrHijo[i] = arrPadre[i];
     // Copiando material genético de la madre
     for (i = k2; i < Constantes.C_LONG_INDIVIDUO; i++)
       arrHijo[i] = arrMadre[i];
   }
   return arrHijo;
 }
```

La función que realiza el *crossover* uniforme se muestra en la figura 31. Ésta función recibe los siguientes parámetros:

- arrPadre: arreglo de una dimensión que será el padre del nuevo individuo.
- arrMadre: arreglo de una dimensión que será la madre del nuevo individuo.
- strPatron: cadena binaria (ceros y unos) que se tomará como patrón para elegir una posición del padre o de la madre al momento de realizar el cruce.
- intHijo: número de hijo a generar. Indica la forma en que se elegirá al padre o la madre dependiendo del patrón, puede ser seleccionar una posición del padre cuando el patrón es 1 y una posición de la madre cuando el patrón es 0 o viceversa.

Figura 31. Función de *crossover* uniforme

```
public float[] crossoverUniforme(float arrPadre[], float arrMadre[], String strPatron,
                                 int intHijo) {
    float arrHijo[] = new float[Constantes.C_LONG_INDIVIDUO];
    int i;

    if (intHijo == 1) {
      for (i = 0; i<Constantes.C_LONG_INDIVIDUO; i++) {
        //Patrón 1 tomar padre, 0 tomar madre
        if (strPatron.charAt(i) == '1')
          arrHijo[i] = arrPadre[i];
        else if (strPatron.charAt(i) == '0')
          arrHijo[i] = arrMadre[i];
      }
    }
    else if (intHijo == 2) {
      for (i = 0; i<Constantes.C_LONG_INDIVIDUO; i++) {
        //Patrón 1 tomar madre, 0 tomar padre
        if (strPatron.charAt(i) == '1')
          arrHijo[i] = arrMadre[i];
        else if (strPatron.charAt(i) == '0')
          arrHijo[i] = arrPadre[i];
      }
    }
    return arrHijo;
  }
```

La función que realiza el *crossover* de dos puntos asimétrico se muestra en la figura 32. Ésta función recibe los siguientes parámetros:

- arrPadre: arreglo de una dimensión que será el padre del nuevo individuo.
- arrMadre: arreglo de una dimensión que será la madre del nuevo individuo.
- k1: posición inicial para intercambio en individuo padre.
- k2: posición final para intercambio en individuo padre.
- k3: posición inicial para copia en individuo madre.
- intHijo: número de hijo a generar. Indica la forma en que se generará al hijo, puede ser mediante reemplazar material genético en el padre o en la madre.

Figura 32. Función de *crossover* de dos puntos asimétrico

```
public float[] atc_pos(float arrPadre[], float arrMadre[], int k1,
                                  int k2, int k3, int intHijo) {
   float arrHijo[] = new float[Constantes.C_LONG_INDIVIDUO];
   int i,j;

   if (intHijo == 1) {
     for (i=0; i<Constantes.C_LONG_INDIVIDUO; i++)
       arrHijo[i] = arrPadre[i];
     j = k3;
     // Reemplazar material genético del padre entre k1 y k2 con material
     // genético de la madre a partir de k3
     for (i = k1; i <= k2; i++) {
       arrHijo[i] = arrMadre[j];
       // Si se llegó al final del arreglo madre, iniciar en 0 (circular)
       if (j == Constantes.C_LONG_INDIVIDUO - 1)
         j = 0;
       else
         j = j + 1;
     }
   }
   else if (intHijo == 2) {
     for (i=0; i<Constantes.C_LONG_INDIVIDUO; i++)
       arrHijo[i] = arrMadre[i];
     // Copiar material genético del padre entre k1 y k2 en el arreglo
     // madre entre k1 y k2
     for (i = k1; i <= k2; i++)
       arrHijo[i] = arrPadre[i];
   }
   return arrHijo;
 }
```

La función que realiza la mutación recibe como parámetro un arreglo de una dimensión que es el individuo a mutar, y cambia una de sus posiciones por un factor numérico aleatorio, de la misma forma que se genera un factor en la población inicial. Ésta función se muestra en la figura 33.

Figura 33. Función de mutación

```
public float[] mutacion(float arrIndividuo[]) {
    float arrNuevo[] = new float[Constantes.C_LONG_INDIVIDUO];
    int k;
    float fltNuevoFactor, fltRango;

    // Calculando posición a reemplazar
    k = (int) Math.floor(Constantes.C_LONG_INDIVIDUO * Math.random());

    // Calculando nuevo factor al azar
    fltRango = Constantes.C_FACTOR_MAX - Constantes.C_FACTOR_MIN;
    // Calcular factor en base al rango
    fltNuevoFactor = Constantes.C_FACTOR_MIN +
                      (fltRango * new Double(Math.random()).floatValue());

    // Reemplazar factor del individuo original
    arrNuevo = arrIndividuo;
    arrNuevo[k] = fltNuevoFactor;

    return arrNuevo;
  }
```

5. EXPERIMENTACIÓN

5.1. Procedimiento y equipo

El proceso de experimentación consistirá en probar la aplicación con un conjunto de datos y determinados parámetros y almacenar los resultados de cada ejecución. Los resultados de cada ejecución se escriben a un archivo, donde se almacena la hora de inicio y fin del experimento, y un registro del mejor individuo con su punteo para cada generación. Un ejemplo de la salida del archivo se muestra en la figura 34.

Figura 34. Ejemplo de la salida a archivo de la aplicación

```
Inicia ejecución de algoritmo. Fecha: Thu Jan 26 10:32:04 GMT-06:00 2006
Generación: 1
Mejor individuo: [-33.493645, 42.955124, -2.443779, -97.50764, 20.124947, 52.49565, -
65.26201, -49.85375, 7.0184555, -0.115119934]
Mejor punteo: 5807.563993019705
5807.563993019705
Generación: 2
Mejor individuo: [-63.649223, -49.724113, -97.50775, 7.788704, -81.57228, 75.136505, -
7.7783585, 99.261215, -52.75114, 1.4263153]
Mejor punteo: 12268.978769291869
12268.978769291869
Generación: 3
Mejor individuo: [-63.649223, -49.724113, -97.50775, 7.788704, 41.186386, 75.136505, -
94.04253, 99.261215, -52.75114, 1.4263153]
Mejor punteo: 12457.790480423562
12457.790480423562

…
Generación: 34500
Mejor individuo: [99.99545, 99.99901, 99.99802, 99.99916, 99.99991, 99.99913, 99.99983,
99.99994, -73.72555, 2.0792694]
Mejor punteo: 16796.45102100853
Ejecución finalizada. Fecha: Thu Jan 26 12:53:45 GMT-06:00 2006
```

Se ejecutará el algoritmo varias veces con los mismos parámetros para comprobar su comportamiento estocástico, es decir, que en varias ejecuciones bajo las mismas condiciones se obtengan resultados distintos.

Luego se variarán los parámetros y se compararán los resultados obtenidos, para llegar a diversas conclusiones sobre el funcionamiento del algoritmo genético desarrollado.

Se manejarán baterías de experimentos que trabajarán con el mismo conjunto de datos. Cada batería estará formada por varios grupos de experimentos donde cada grupo se ejecutará con los mismos parámetros.

Todos los experimentos se realizarán en una computadora con procesador Intel Pentium 4 de 2.8 Gigahertz y 512 Megabytes de memoria RAM.

5.2. Primer batería de experimentos

5.2.1. Conjunto de datos

El primer grupo de experimentos utiliza un conjunto de datos cuyo modelo matemático es conocido, es decir, se sabe cuál es la función que representa los datos. Esto permite evaluar de una mejor manera el comportamiento del algoritmo. Este conjunto de datos consta de 36 periodos de tiempo con un valor constante igual a 100 como se muestra en la tabla VIII.

Tabla VIII. Conjunto de datos para el primer grupo de experimentos

Período de tiempo	Valor (constante)
1	100
2	100
3	100
4	100
5	100
…	…
36	100

Se sabe que el modelo matemático que se ajusta exactamente al conjunto de datos de prueba está dado por una constante, como se muestra a continuación:

$$Y = 100$$

El modelo matemático se encuentra cuando el primer factor numérico es igual a 100 y todos los otros factores son iguales a 0, de la siguiente manera:

$$Y = 100 \cdot X^0 + 0 \cdot X^1 + 0 \cdot X^2 + \ldots + 0 \cdot X^n$$

Así, el mejor individuo de la población es un arreglo cuya primera posición sea igual a 100 y el resto de posiciones sean 0: [100, 0, 0,…, 0]

5.2.2. Diseño del experimento

Se ejecutarán cuatro grupos de tres experimentos cada uno. Cada grupo se ejecutará con los mismos parámetros y se modificarán para crear un nuevo grupo de experimentos. En total se ejecutarán nueve experimentos, como se muestra en la tabla IX.

Tabla IX. Diseño de la primer batería de experimentos

	Experimento		
Grupo	A	B	C
1	Experimento 1A Parámetros 1	Experimento 1B Parámetros 1	Experimento 1C Parámetros 1
2	Experimento 2A Parámetros 2	Experimento 2B Parámetros 2	Experimento 2C Parámetros 2
3	Experimento 3A Parámetros 3	Experimento 3B Parámetros 3	Experimento 3C Parámetros 3
4	Experimento 4A Parámetros 4	Experimento 4B Parámetros 4	Experimento 4C Parámetros 4

Se utiliza el mismo conjunto de parámetros para más de un experimento porque se busca comprobar el comportamiento estocástico del algoritmo genético, es decir si se obtienen diferentes resultados bajo las mismas condiciones. Además se busca identificar un grupo de parámetros adecuado para los experimentos posteriores.

La primera configuración de parámetros, para los experimentos 1A, 1B y 1C es la siguiente:

- Grado del polinomio: 10
- Factor numérico mínimo: -100
- Factor numérico máximo: 100
- Tamaño de la población: 1000
- Tipo de *crossover*: Un punto
- Mutaciones: 500
- Mutar al mejor individuo: si
- Tipo de función de evaluación: DAM (Desviación Absoluta de la Media)
- Divisor para la función de evaluación: 1000000
- Tipo de selección: Elitista
- Número de individuos mejores a elegir: 500
- Condición de finalización: Una hora aproximadamente.

La segunda configuración de parámetros, para los experimentos 2A, 2B y 2C es la siguiente:

- Grado del polinomio: 10
- Factor numérico mínimo: -100
- Factor numérico máximo: 100
- Tamaño de la población: 1000

- Tipo de *crossover*: Dos puntos
- Mutaciones: 500
- Mutar al mejor individuo: si
- Tipo de función de evaluación: EMC (Error Medio Cuadrado)
- Divisor para la función de evaluación: 1000000
- Tipo de selección: Elitista
- Número de individuos mejores a elegir: 500
- Condición de finalización: 2 horas aproximadamente ó 34500 ejecuciones.

La tercera configuración de parámetros, para los experimentos 3A, 3B y 3C es la siguiente:

- Grado del polinomio: 10
- Factor numérico mínimo: -100
- Factor numérico máximo: 100
- Tamaño de la población: 1000
- Tipo de *crossover*: Uniforme
- Patrón para *crossover* uniforme: 1010101010
- Mutaciones: 500
- Mutar al mejor individuo: si
- Tipo de función de evaluación: PEMA (Porcentaje de Error Medio Absoluto)
- Divisor para la función de evaluación: 1000000
- Tipo de selección: Elitista
- Número de individuos mejores a elegir: 500
- Condición de finalización: 34500 ejecuciones.

La cuarta configuración de parámetros, para los experimentos 4A, 4B y 4C es la siguiente:

- Grado del polinomio: 10
- Factor numérico mínimo: -100
- Factor numérico máximo: 100
- Tamaño de la población: 1000
- Tipo de *crossover*: Uniforme
- Patrón para *crossover* uniforme: 1100110011
- Mutaciones: 800
- Mutar al mejor individuo: si
- Tipo de función de evaluación: PEMA (Porcentaje de Error Medio Absoluto)
- Divisor para la función de evaluación: 1000000
- Tipo de selección: Generacional
- Número de individuos mejores a elegir: 500
- Condición de finalización: 34500 ejecuciones.

5.2.3. Resultados

5.2.3.1 Primer grupo de experimentos

Experimento 1A

- Tiempo de ejecución: 57 minutos
- Número de ejecuciones: 9714
- Mejor punteo: 19648.020480762454
- Mejor individuo: [99.9706, 99.99556, 99.967606, 83.63507, -34.412476, -94.67515, -99.40836, -98.890015, 7.2013016, -0.12365723]

Experimento 1B

- Tiempo de ejecución: 57 minutos
- Número de ejecuciones: 9758
- Mejor punteo: 86955.51386721047
- Mejor individuo: [92.21326, 99.999176, 99.99997, 99.99992, 99.99959, 99.99774, 99.99939, 99.99968, -10.181747, 0.21035767]

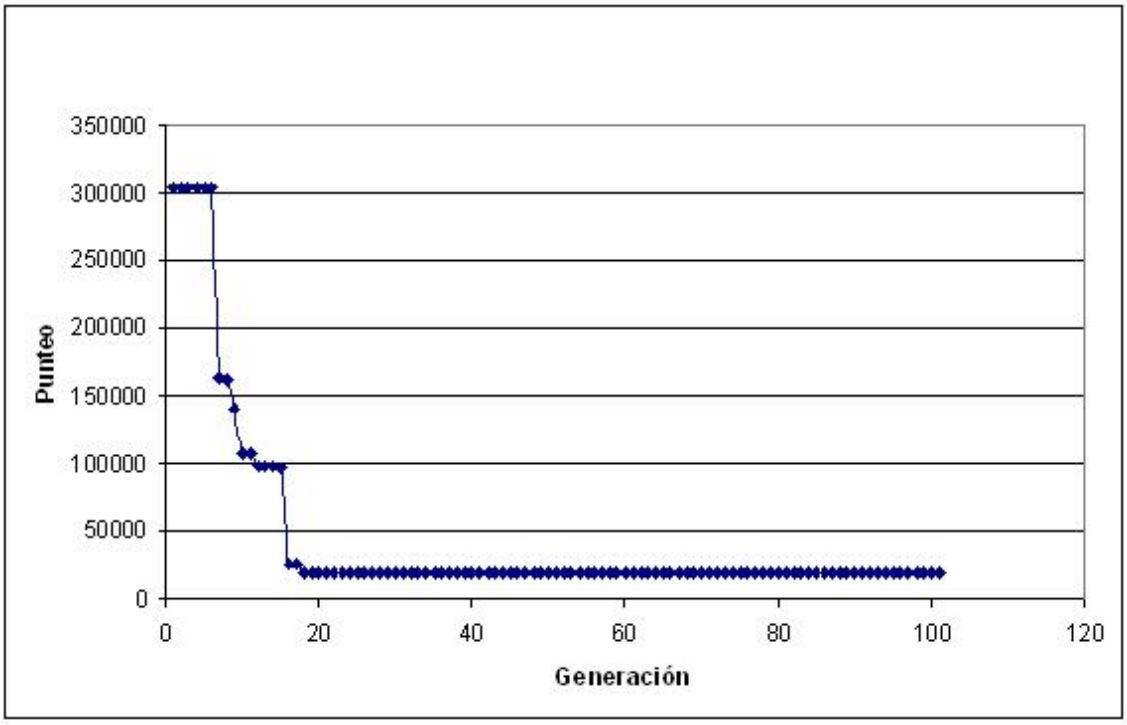
Experimento 1C

- Tiempo de ejecución: 57 minutos
- Número de ejecuciones: 9858
- Mejor punteo: 13508.868197221487
- Mejor individuo: [-99.99934, -99.997, -3.042015, 99.99823, 99.99805, 99.99872, 96.56363, 55.941467, -4.3177032, 0.07606506]

Se puede observar en este primer grupo que los mejores individuos para cada experimento son distintos y por tanto los punteos para cada uno también son distintos. Además, el mejor individuo es el del experimento 1C pues tiene el menor punteo, que es de 13508.87.

Se analizarán ahora más detalladamente los resultados del mejor experimento, el 1C. La figura 35 muestra los punteos encontrados para las primeras 100 generaciones, puede observarse una gran mejora durante las primeras generaciones.

Figura 35. Punteos para las primeras cien generaciones del experimento 1C

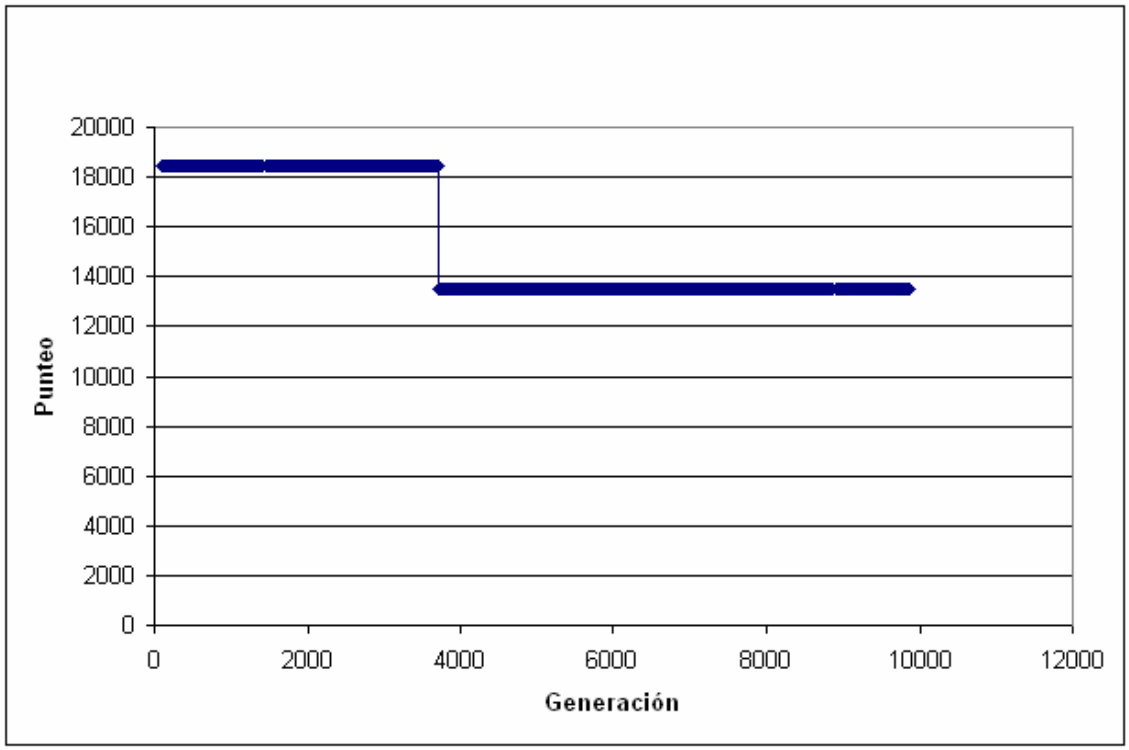


En la ejecución número 100 el mejor punteo fue de 18469.8699, aunque es superior al punteo de la última ejecución, la mejora no es significativa.

Si se quitan las primeras cien generaciones, se obtiene la gráfica de la figura 36. Aquí se han graficado los punteos de la generación 100 a la 9858. Puede observarse sólo una mejora significativa que se produce al pasar de la generación 3698 con un punteo de 18462.5285 a la generación 3699 con un punteo de 13539.0026.

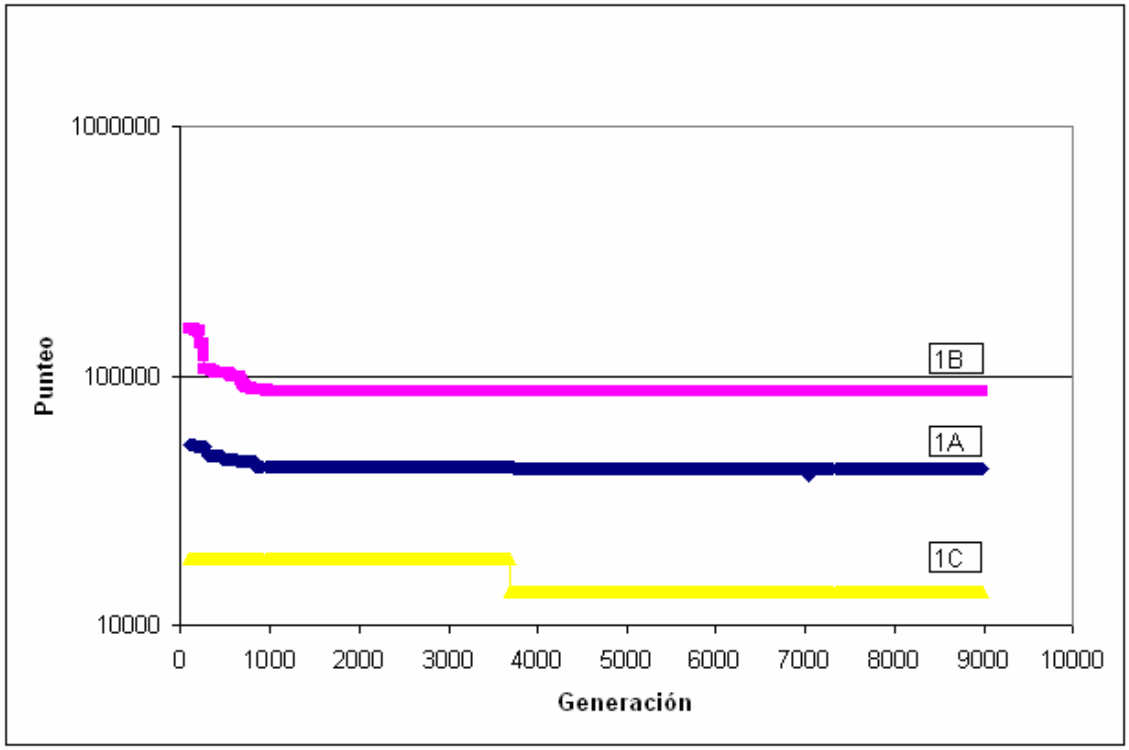
El punteo obtenido en la última generación es de 13508.8682, lo cual indica que luego de la generación 3699 los cambios fueron mínimos. Estos cambios no pueden apreciarse en la gráfica.

Figura 36. Punteos para el experimento 1C



Una comparación de los resultados obtenidos en los tres experimentos de este primer grupo, 1A, 1B y 1C, se muestra en la figura 37. Puede verse que los tres experimentos se comportaron de manera distinta, aunque trabajaron con los mismos parámetros. El experimento 1C presentó los mejores punteos, pues son menores a los del experimento 1A y 1B. También se puede observar que los tres experimentos convergen en un punto donde se mantiene casi constante el punteo de los mejores individuos de cada generación.

Figura 37. Comparación de los resultados obtenidos en el primer grupo de experimentos, 1A, 1B y 1C



5.2.3.2 Segundo grupo de experimentos

Experimento 2A

- Tiempo de ejecución: 2 horas y 3 minutos
- Número de ejecuciones: 34524
- Mejor punteo: 1.6304309492985325E15
- Mejor individuo: [-99.99953, -99.9999, -99.99999, -99.99996, -99.999954, -99.99942, -99.99898, -98.18771, 7.2237396, -0.1242218]

Luego de haber realizado este experimento, se determinó un número de ejecuciones para los siguientes experimentos de 34500, el cual lleva aproximadamente un tiempo de 2 horas.

Experimento 2B

- Tiempo de ejecución: 2 horas y 25 minutos
- Número de ejecuciones: 34500
- Mejor punteo: 1.4009522575186736E17
- Mejor individuo: [99.99997, 99.99994, 99.999985, 99.99925, 99.999954, 99.99982, 99.99997, 99.99989, -15.33947, 0.358284]

Experimento 2C

- Tiempo de ejecución: 2 horas y 25 minutos
- Número de ejecuciones: 34500
- Mejor punteo: 3.193500064553921E15
- Mejor individuo: [99.99834, 99.99954, 99.99997, 99.99913, 99.99951, 99.99997, 99.99866, 99.99919, -7.750313, 0.13785553]

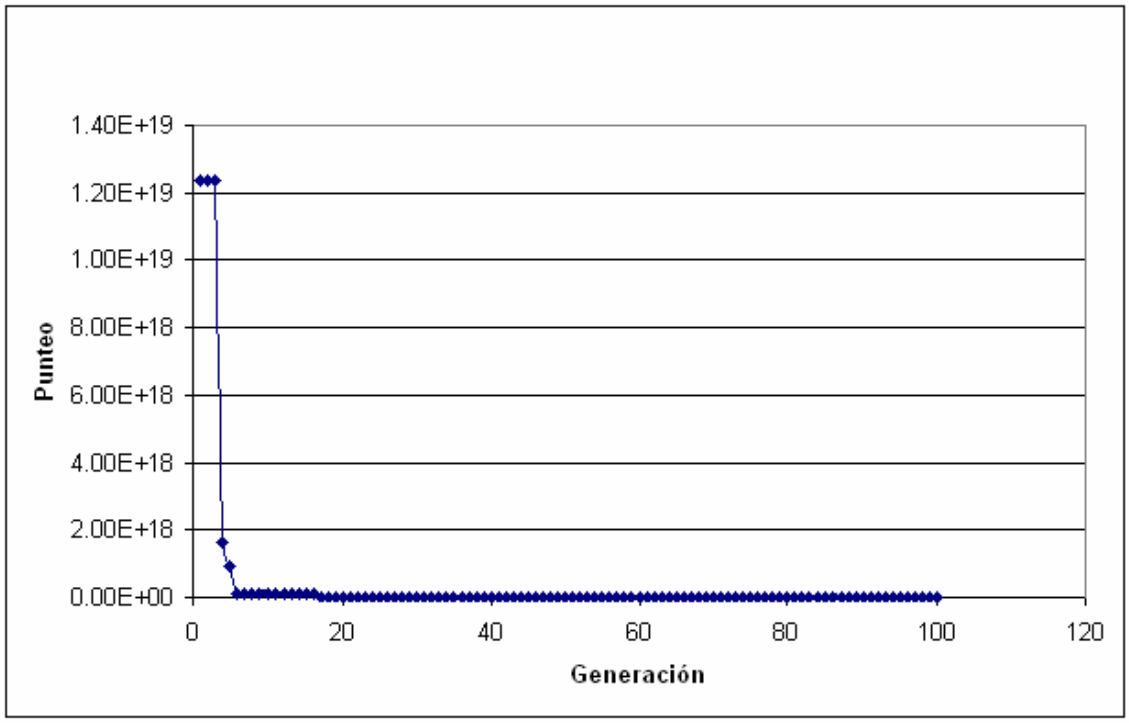
De este grupo de experimentos nuevamente puede verse que los resultados obtenidos son distintos y el mejor experimento fue el 2A, con el menor punteo de 1.6304309492985325E15. Los punteos hallados en éste grupo son mayores a los del primer grupo debido a que se modificó la función de evaluación utilizada, y la nueva función, EMC error medio cuadrado, eleva las diferencias al cuadrado.

Analizando más detenidamente los resultados del experimento 2A, tenemos en la figura 38 los punteos para las primeras 100 generaciones.

Aquí puede observarse el mismo comportamiento que en el experimento 1C, el punteo de los individuos mejora bastante durante las primeras generaciones y luego se mantiene casi constante.

El mejor punteo para la generación 100 es de 9.616262843292438E15, y comparándolo con el punteo encontrado al final del experimento puede decirse que existe cierta mejora significativa al continuar su ejecución.

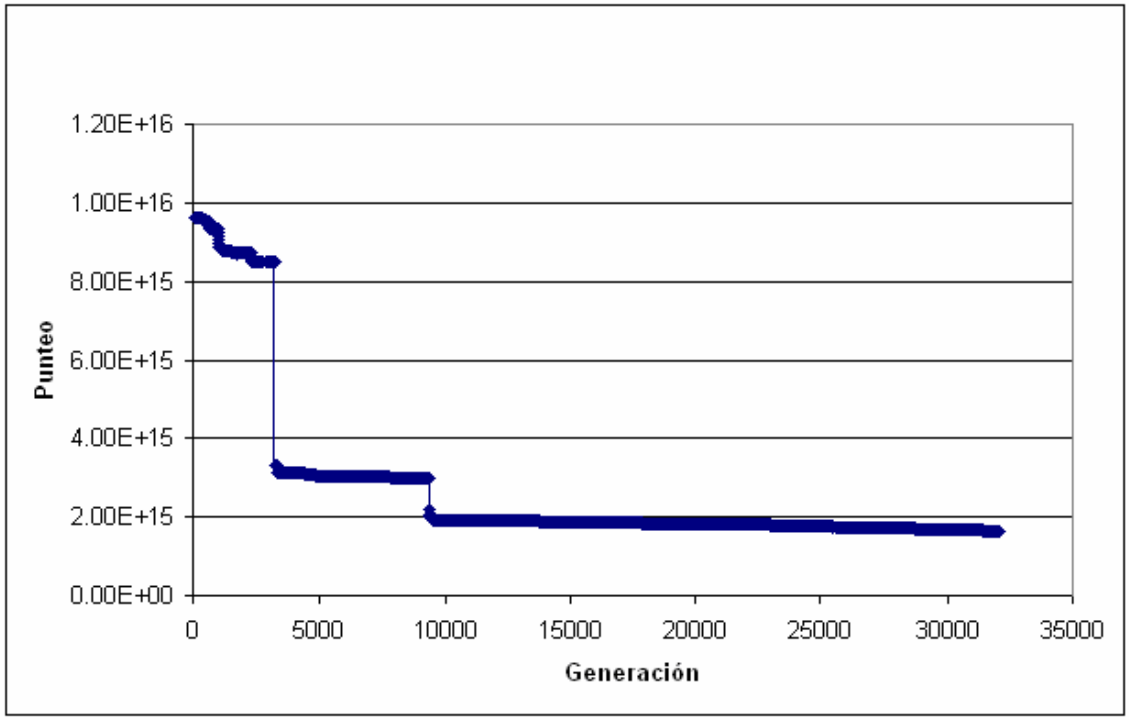
Figura 38. Punteos para las primeras cien generaciones del experimento 2A



Si se omiten las primeras 100 generaciones, la gráfica de la generación 100 a la 34524 se muestra en la figura 39.

Aquí pueden observarse dos cambios significativos, el primero se produce entre la generación 3253 con un punteo de 8.484790852472101E15 y la generación 3254 con un punteo de 3.320825865327562E15. El segundo cambio significativo se encuentra entre la generación 9631 con un punteo de 1.9323998195681182E15 y la generación 9632 con un punteo de 1.9322306590266002E15.

Figura 39. Punteos para el experimento 2A



Se debe tomar en cuenta que en este experimento se observan cambios más significativos debido a que los punteos obtenidos con la función de evaluación también son mayores.

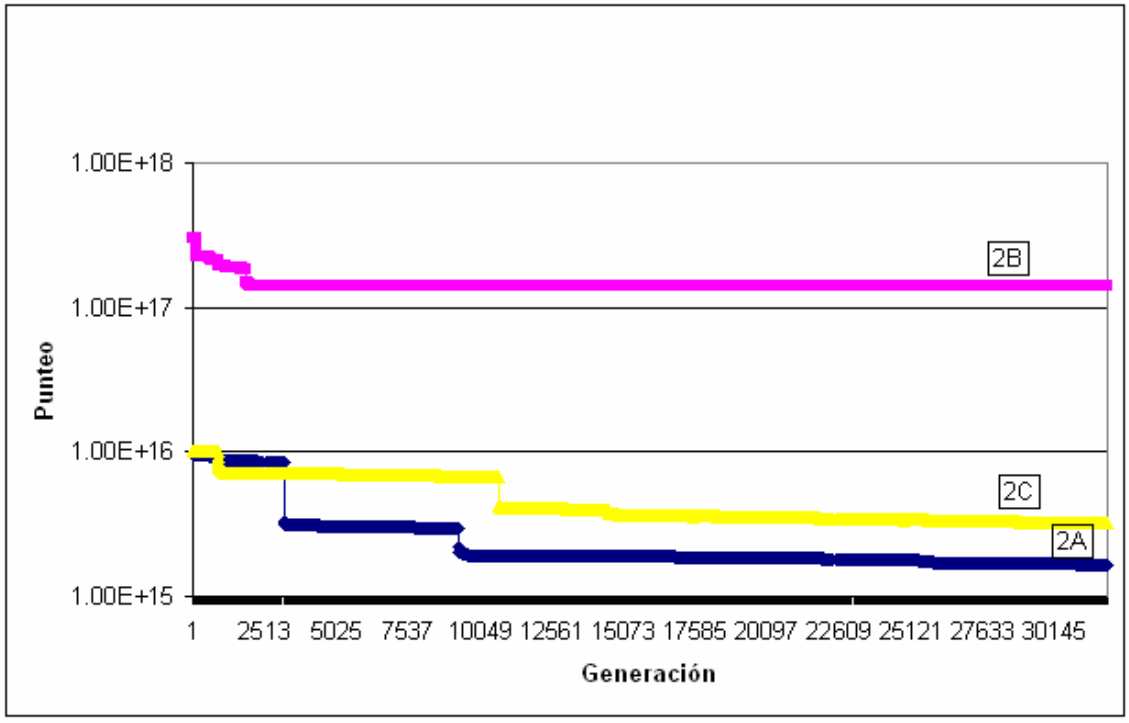
Puede notarse que al finalizar el experimento, existía cierta tendencia hacia abajo aunque sin cambios significativos. El mejor individuo en la generación 10000 fue:

[-99.99779, -99.99874, -99.99848, -99.99996, -99.99905, -99.99836, -99.98323, -95.53216, 7.155464, -0.12451172]

Que es muy parecido al mejor individuo de la última generación, por lo que puede decirse que el algoritmo encontró un punto de convergencia aproximadamente a partir de la generación 10000.

Una gráfica comparativa de los resultados de los tres experimentos del segundo grupo se presenta en la figura 40. Puede observarse en esta gráfica que el experimento 2A presenta los menores punteos, lo que hace que sea el mejor de este grupo. Nuevamente se puede comprobar aquí el comportamiento estocástico del algoritmo.

Figura 40. Comparación de los resultados obtenidos en el segundo grupo de experimentos, 2A, 2B y 2C



5.2.3.3 Tercer grupo de experimentos

En este grupo de experimentos además se registró el número de generación que obtuvo el mejor punteo durante toda la ejecución del experimento.

Experimento 3A

- Tiempo de ejecución: 2 horas y 21 minutos
- Número de ejecuciones: 34500
- Mejor punteo de la última ejecución: 16796.45102100853
- Mejor individuo: [99.99545, 99.99901, 99.99802, 99.99916, 99.99991, 99.99913, 99.99983, 99.99994, -73.72555, 2.0792694]
- Mejor generación: 19
    - Punteo: 763.243492548329
    - [-68.071106, -53.635075, -3.4399414, 64.66133, 76.26135, 52.51358, -2.3441162, 16.434113, 1.8164902, -0.070121765]

Experimento 3B

- Tiempo de ejecución: 2 horas y 21 minutos
- Número de ejecuciones: 34500
- Mejor punteo de la última ejecución: 3183.130781854272
- Mejor individuo: [-99.99553, -99.99996, -99.999756, -99.99966, -99.99988, -99.999664, -99.999985, -99.9996, 19.419373, -0.4820404]
- Mejor generación: 15070
    - Punteo: 2252.7901884065996
    - Individuo [-99.997375, -99.99975, -99.999756, -99.99966, -99.99813, -99.999664, -99.99985, -99.99914, -0.40498352, 0.114715576]

Experimento 3C

- Tiempo de ejecución: 1 hora y 59 minutos
- Número de ejecuciones: 34500
- Mejor punteo de la última ejecución: 12180.493973693941
- Mejor individuo: [99.96033, 99.999664, 99.99855, 99.99959, 99.99983, 99.999176, 99.99986, 99.99983, -55.3118, 1.5376892]

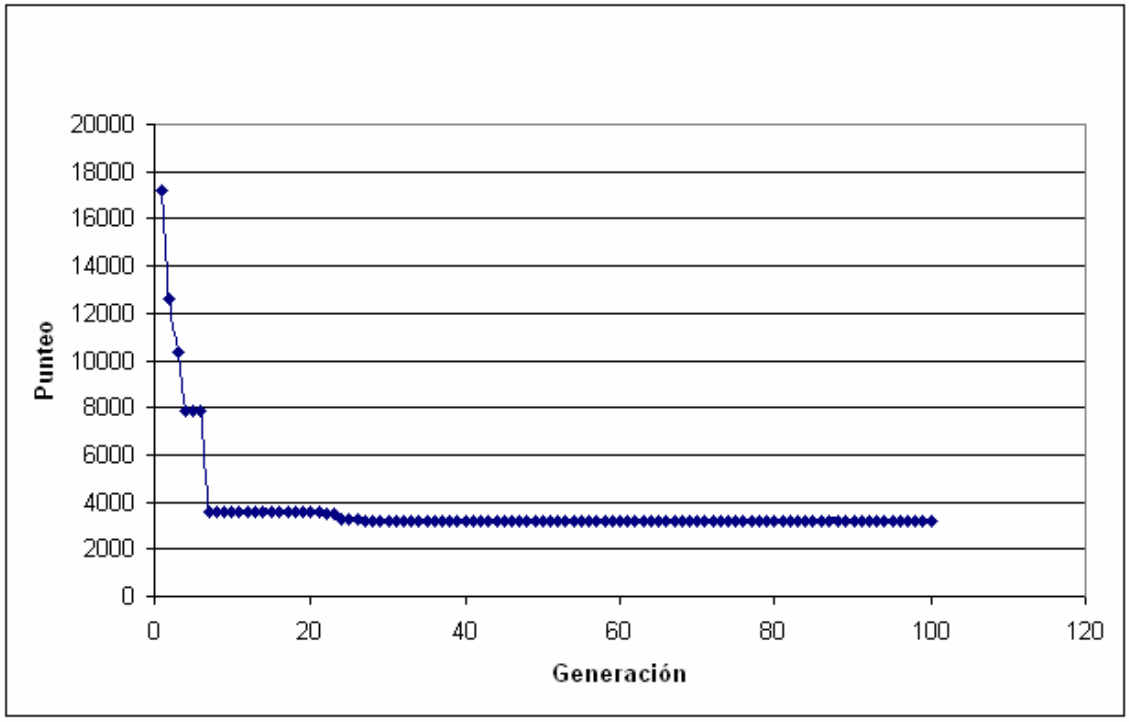
- Mejor generación: 7666
    - Punteo: 338.42790607646657
    - [80.77455, 99.99719, -70.76176, -14.673859, 99.997345, 99.999176, 99.99986, 50.667114, -2.2677078, 0.01952362]

Se puede ver que en estos experimentos el algoritmo genético encontró un mejor punto mucho antes de llegar al número total de ejecuciones, lo cual indica que con los parámetros actuales no se garantiza que un mayor tiempo de ejecución vaya a dar mejores resultados, esto se debe a que las mutaciones se aplican a toda la población lo cual evita que el mejor individuo se mantenga de una generación a la siguiente.

Las mutaciones son buenas porque introducen diversidad en la población, sin embargo, mutar al mejor individuo de cada generación también puede ocasionar que no se llegue a un punto óptimo, como ocurrió en este grupo de experimentos. Para evitar esto pueden modificarse los parámetros para que no se realicen mutaciones sobre el mejor individuo de cada generación, garantizando su existencia en las generaciones futuras y reemplazándolo sólo por un individuo mejor.

El mejor experimento de este grupo, tomando en cuenta el punteo obtenido en la última ejecución, fue el 3B. Una gráfica de las primeras 100 generaciones de éste experimento se muestra en la figura 41.

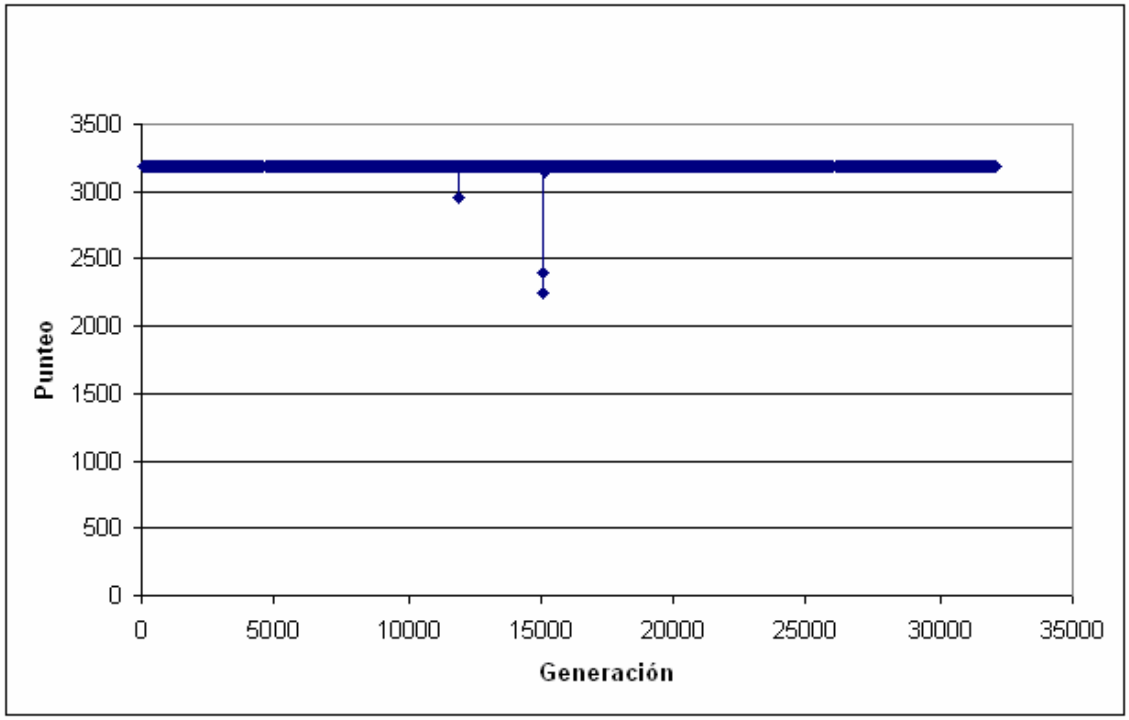
Figura 41. Punteos para las primeras cien generaciones del experimento 3B



Nuevamente se observa el mismo comportamiento al inicio, un cambio drástico en el punteo de las primeras generaciones. Los punteos aquí son los más bajos de los tres grupos vistos hasta el momento, esto se debe a que la función de evaluación utilizada, PEMA porcentaje de error medio absoluto, toma valores relativos (porcentajes) en vez de valores absolutos.

La gráfica del punteo del resto de generaciones, generación 100 a la 34500, se muestra en la figura 42.

Figura 42. Punteos para el experimento 3B

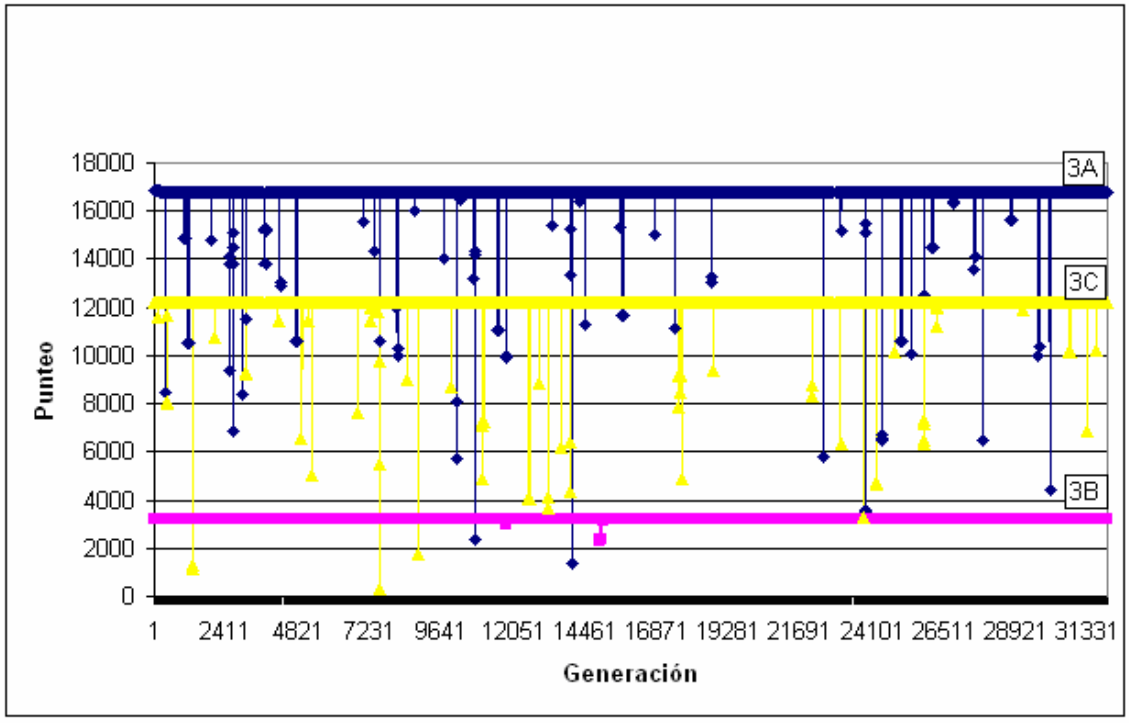


Aquí se observan dos puntos significativamente mejores del resto, pero que no mejoran la calidad de individuos de las siguientes generaciones. El primero se encuentra en a partir de la generación 11888 y se mantiene por 12 generaciones consecutivas, aunque no se aprecia en la gráfica debido a la escala utilizada, pero luego el puntaje de las siguientes generaciones empeora. El segundo punto mejor se encuentra en la generación 15070 y se mantiene hasta la generación 15088, para luego regresar a individuos con peor puntaje.

Esto puede evitarse si se garantiza siempre la sobre vivencia del individuo más apto asegurando que no sea afectado por las mutaciones. De esta forma al llegar al primer punto significativamente mejor, en la generación 11888, se hubiera garantizado que el mejor individuo de ésta generación se mantuviera hasta el siguiente punto mejor en la generación 15070 y era probable que al final de la ejecución se encontrara aún un individuo con mejor punteo.

La gráfica comparativa de los resultados obtenidos en este tercer grupo de experimentos, 3A, 3B y 3C se muestra en la figura 43. Aquí puede observarse que existen muchas variaciones durante la ejecución de los experimentos 3A y 3C, mientras que el experimento 3B se mantiene casi constante. Puede observarse que existieron muchos puntos mejores que los hallados al final de cada experimento, lo cual puede evitarse si se restringen las mutaciones para que no afecten al mejor individuo de cada generación.

Figura 43. Comparación de resultados obtenidos en el tercer grupo de experimentos, 3A, 3B y 3C



5.2.3.4 Cuarto grupo de experimentos

Experimento 4A

- Tiempo de ejecución: 3 horas y 27 minutos
- Número de ejecuciones: 34500

- Mejor punteo de la última ejecución: 553.3956803436075
- Mejor individuo: [55.56372, -99.62887, -99.94884, -99.92386, -99.998085, -99.99956, -99.99907, -91.22588, 8.346558, -0.16396332]
- Mejor generación: 14
    - Punteo: 151.8375197372875
    - [-62.059723, 60.76796, 94.293945, -5.2072372, 30.223282, 5.8569946, 9.635582, 32.754593, -2.704094, 0.050689697]

Experimento 4B

- Tiempo de ejecución: 4 horas y 32 minutos
- Número de ejecuciones: 34500
- Mejor punteo de la última ejecución: 464.1839464279133
- Mejor individuo: [-82.769226, 98.03041, 99.977005, 99.98384, 99.99806, 99.99956, 99.99965, 74.439606, -6.9198227, 0.13652802]
- Mejor generación: 29851
    - Punteo: 257.46357414803117
    - Individuo [-90.592766, 58.8042, 99.977005, 99.98384, 99.994354, 99.99956, 99.99965, 90.503845, -6.9198227, 0.12132263]

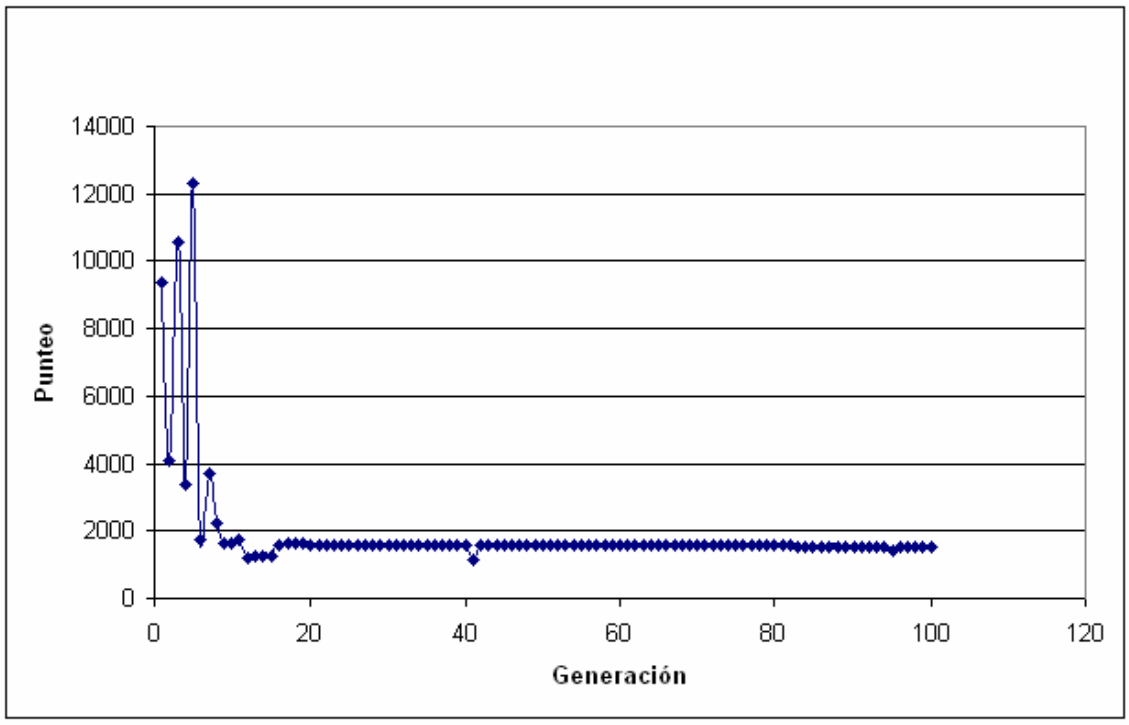
Experimento 4C

- Tiempo de ejecución: 3 horas y 33 minutos
- Número de ejecuciones: 34500
- Mejor punteo de la última ejecución: 2225.576023228061
- Mejor individuo: [-55.360508, -99.651764, -99.90592, -99.996506, -99.99607, -99.999596, -99.998985, -99.99844, 15.599014, -0.3696823]
- Mejor generación: 30102
    - Punteo: 1272.2262836487334
    - [4.9480896, -92.30971, -99.90592, -99.996506, -99.99607, -99.999596, -99.998985, -99.99844, 3.8769684, -0.032066345]

El mejor experimento de éste grupo, comparando los mejores punteos de las últimas ejecuciones, es el 4B que presenta un menor punteo igual a 464.183946. Puede observarse que se alcanzó el mejor punteo de cada experimento antes de llegar a la última ejecución, sobre todo en el experimento 4A donde se alcanzó el mejor punteo en la generación 14, apenas al iniciar las ejecuciones.

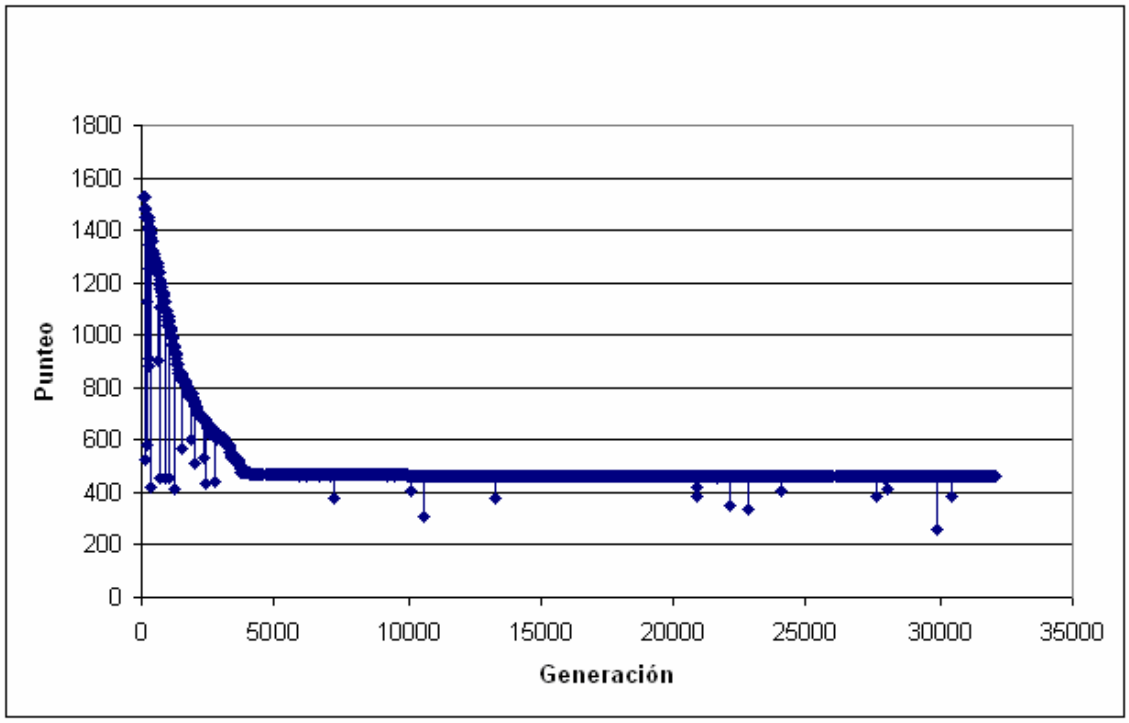
Una gráfica de las primeras cien generaciones del experimento 4B, que fue el mejor de este grupo, se muestra en la figura 44. Puede observarse un comportamiento variable al inicio, pero alcanza un punto estable aproximadamente luego de la generación número veinte.

Figura 44. Punteos para las primeras cien generaciones del experimento 4B



Si se grafican los punteos obtenidos para la generación cien en adelante, se obtiene la gráfica mostrada en la figura 45. Puede verse que sigue existiendo cierta variabilidad en los punteos, aunque ésta disminuye cerca de la generación número cinco mil. Si se compara este resultado con el de los primeros tres grupos de experimentos, se puede observar que este experimento tardó más en converger en un punto pero obtuvo también un mejor individuo.

Figura 45. Punteos para el experimento 4B

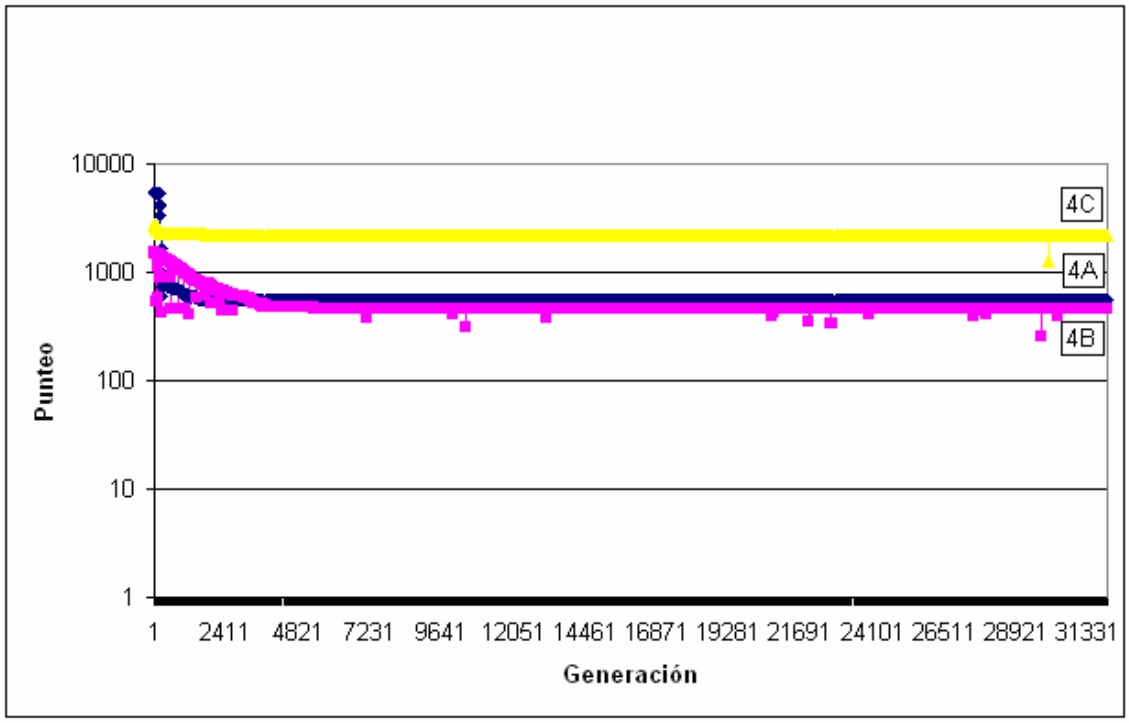


La figura 46 muestra una gráfica comparativa de los punteos obtenidos en los experimentos de este cuarto grupo, 4A, 4B y 4C. Puede observarse que el experimento 4B presenta los menores punteos, seguido de cerca por el experimento 4A, mientras el experimento 4C presenta punteos mayores.

Puede verse que los resultados de este cuarto grupo de experimentos son mejores a los del tercer grupo, pues sus punteos son menores. Esta mejora se logró realizando los siguientes cambios en los parámetros:

- Patrón para *crossover* uniforme modificado de 1010101010 (tercer grupo) a 1100110011 (cuarto grupo)
- Número de mutaciones incrementado de 500 (tercer grupo) a 800 (cuarto grupo)
- Tipo de selección modificado de elitista (tercer grupo) a generacional (cuarto grupo)

Figura 46. Comparación de los resultados obtenidos en el cuarto grupo de experimentos, 4A, 4B y 4C



5.3. Segunda batería de experimentos

5.3.1. Conjunto de datos

Para esta batería de experimentos se han empleado datos reales que corresponden a estadísticas de exportación de café en general que realiza Guatemala.

Para el experimento se han tomado 36 periodos mensuales, correspondientes a las exportaciones de café entre enero del 2001 y diciembre del 2003. Los datos están expresados en miles de quintales y se muestran en la tabla X. Esta información ha sido recolectada por la Asociación Nacional del Café ANACAFE y están disponibles en el sitio Web del banco de Guatemala. [17]

Tabla X. Exportaciones de café de Guatemala

Año	Mes	Miles de quintales	Año	Mes	Miles de quintales
2001	Enero	549.9	2002	Julio	429.6
2001	Febrero	648	2002	Agosto	368.9
2001	Marzo	711.3	2002	Septiembre	403.9
2001	Abril	666.6	2002	Octubre	205.5
2001	Mayo	660.3	2002	Noviembre	174.4
2001	Junio	620.3	2002	Diciembre	274.9
2001	Julio	421	2003	Enero	381.3
2001	Agosto	295	2003	Febrero	503.5
2001	Septiembre	244.1	2003	Marzo	609.1
2001	Octubre	104.6	2003	Abril	655.1
2001	Noviembre	120.6	2003	Mayo	565
2001	Diciembre	241.1	2003	Junio	626.6
2002	Enero	318	2003	Julio	467
2002	Febrero	426.3	2003	Agosto	393.5
2002	Marzo	517.7	2003	Septiembre	298.8
2002	Abril	496.3	2003	Octubre	148.3
2002	Mayo	499	2003	Noviembre	142.3
2002	Junio	421.3	2003	Diciembre	190.8

Fuente: Banco de Guatemala. www.banguat.gob.gt

5.3.2. Diseño del experimento

En este experimento, no se conoce nada acerca del modelo matemático que representa los datos, por lo que primero se realizarán ejecuciones de prueba para validar los parámetros adecuados para el algoritmo genético y luego se procederá a ejecutar tres experimentos durante un período prolongado de tiempo, aproximadamente veinticuatro horas, para probar si existe una mejora en el resultado del algoritmo cuando el tiempo es mayor.

Se busca en este segundo conjunto de experimentos probar la convergencia del algoritmo, pues luego de un número grande de ejecuciones éste debería converger en un punto aunque no necesariamente sea el óptimo.

Existirán tres experimentos que se ejecutarán con los mismos parámetros, como se muestra en la tabla XI.

Tabla XI. Diseño de la segunda batería de experimentos

	Experimento		
Grupo	A	B	C
1	Experimento 5A Parámetros 1	Experimento 5B Parámetros 1	Experimento 5C Parámetros 1

La numeración de los experimentos se ha continuado a partir de la numeración utilizada en la primera batería de experimentos.

La configuración de parámetros a utilizar para estos experimentos se ha determinado luego de algunas ejecuciones de prueba para encontrar un grupo de parámetros adecuado, y es:

- Grado del polinomio: 10
- Factor numérico mínimo: -50
- Factor numérico máximo: 50
- Tamaño de la población: 5000
- Tipo de crossover: Dos puntos
- Mutaciones: 1000
- Mutar al mejor individuo: No
- Tipo de función de evaluación: PEMA (Porcentaje de error medio absoluto)
- Divisor para la función de evaluación: 1000000
- Tipo de selección: Generacional
- Número de individuos mejores a elegir: 500
- Condición de finalización: Veinticuatro horas aproximadamente.

Para este conjunto de experimentos se decidió no mutar al mejor individuo de cada generación, para evitar los problemas de variabilidad hallados en los experimentos de la primera batería (sección 5.2.3), esto garantizará que el punteo de los individuos siempre mejore o se mantenga igual.

Como el proceso de selección es generacional, se realizó una modificación a la forma de generar la siguiente población, copiando siempre el mejor individuo de la generación actual a la siguiente y generando el resto mediante operaciones genéticas. Cuando no se mantiene al mejor individuo de cada generación, no se copia ningún individuo de la generación actual a la siguiente.

5.3.3. Resultados

Experimento 5A

- Tiempo de ejecución: 25 horas y 10 minutos
- Número de ejecuciones: 85873
- Mejor punteo de la última ejecución: 10.826467277006973
- Mejor individuo: [49.999985, 49.999954, -49.99997, -49.999985, -49.99999, -49.999992, -49.99999, -6.6298714, 0.7095566, -0.013660431]
- Mejor generación: 84068
    - Punteo: 10.826467277006973
    - [49.999985, 49.999954, -49.99997, -49.999985, -49.99999, -49.999992, -49.99999, -6.6298714, 0.7095566, -0.013660431]

Experimento 5B

- Tiempo de ejecución: 24 horas y 18 minutos
- Número de ejecuciones: 83177
- Mejor punteo de la última ejecución: 464.1839464279133
- Mejor individuo: [-49.999985, 49.999992, 49.99996, 49.999985, 49.999992, 50.0, 49.99996, 48.312286, -4.448738, 0.08646774]
- Mejor generación: 76555
    - Punteo: 94.91413864512272
    - Individuo [-49.999985, 49.999992, 49.99996, 49.999985, 49.999992, 50.0, 49.99996, 48.312286, -4.448738, 0.08646774]

Experimento 5C

- Tiempo de ejecución: 24 horas y 28 minutos
- Número de ejecuciones: 84666
- Mejor punteo de la última ejecución: 85.79079180400035

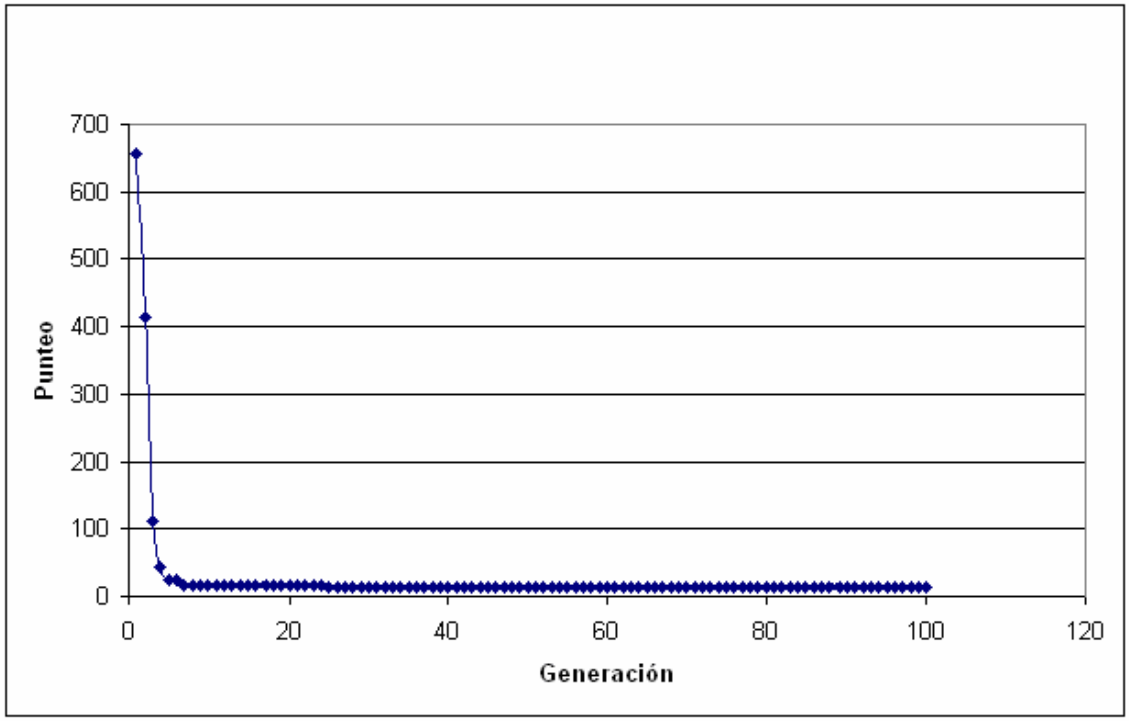
- Mejor individuo: [49.999878, -49.999966, -49.99999, -49.999985, -49.999992, -49.99998, -50.0, -45.975803, 4.1802406, -0.080703735]
- Mejor generación: 84536
    - Punteo: 85.79079180400035
    - [49.999878, -49.999966, -49.99999, -49.999985, -49.999992, -49.99998, -50.0, -45.975803, 4.1802406, -0.080703735]

Se pude observar en los resultados de estos experimentos que el punteo de la mejor generación siempre es igual al punteo de la última generación, esto se debe a que se mantiene siempre el mejor individuo de cada generación, entonces se encuentra una mejor generación sólo cuando existe un individuo con mejor punteo.

La ejecución de estos experimentos fue más lenta que en el primer conjunto debido a que se incrementó el tamaño de la población. Puede observarse que aunque se incrementó el tiempo aproximadamente doce veces (de dos a veinticuatro horas), el número de ejecuciones sólo se incremento cerca de tres veces (de 34500 a 84000).

El mejor experimento de este grupo es el 5A, pues contiene la generación con menor punteo que es de 10.826467277006973. Los punteos de las primeras cien generaciones para el experimento 5A se muestran en la figura 47. Puede observarse una mejora bastante rápida observándose un cambio en el punteo de 657.20 para la primera generación a 24.20 para la quinta generación.

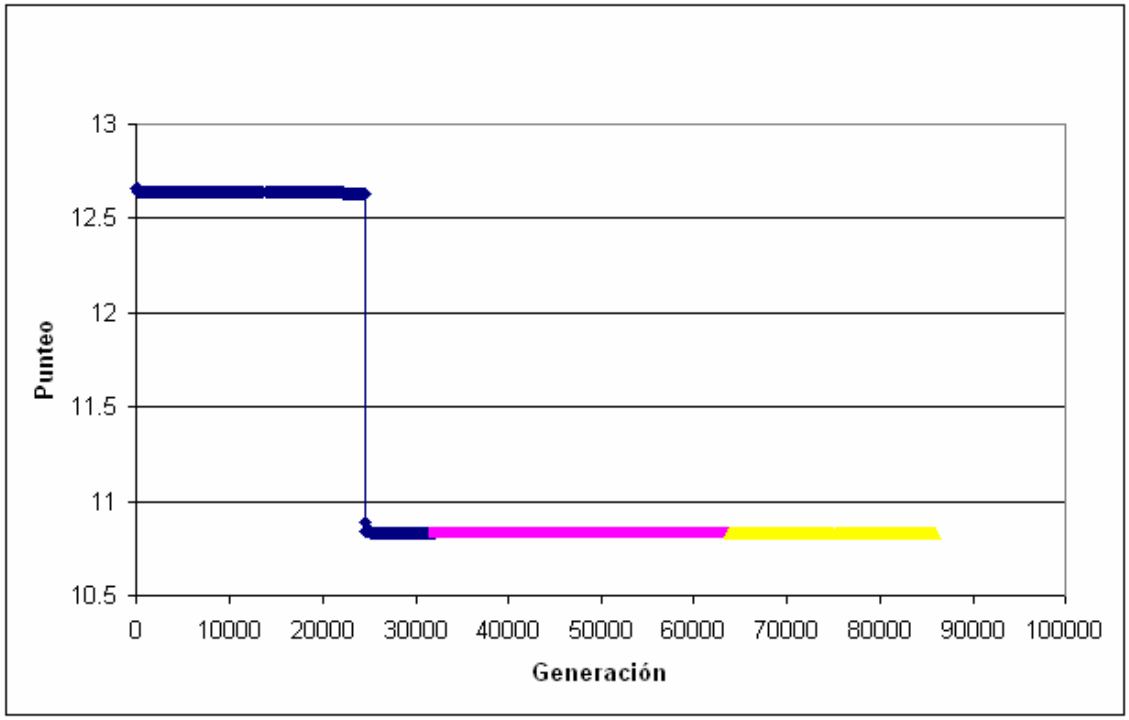
Figura 47. Punteos para las primeras cien generaciones del experimento 5C



Una gráfica de los punteos para toda la ejecución del experimento 5C, exceptuando las primeras cien generaciones, se muestra en la figura 48. La gráfica se ha dividido en tres series debido a la cantidad de puntos involucrados. Puede observarse únicamente un cambio significativo, o al menos visible en la gráfica, que ocurre al pasar de la generación 24621 con un punteo de 12.6351855 a la generación 24622 con un punteo de 10.8894423.

A partir de la generación 24622 los cambios son mínimos, mejorando sólo en 0.0629751 al llegar el final de la ejecución. Además, de los resultados obtenidos en el experimento puede verse que de la generación 84068 a la 85873 (1085 generaciones) no existió ningún cambio.

Figura 48. Punteos para el experimento 5C



Puede concluirse entonces que en el experimento 5C existieron mejoras significativas hasta la generación 24622, lo que representa aproximadamente un 30% del número total de ejecuciones. El individuo encontrado en ésta generación fue:

[49.999985, -49.99998, -49.999966, -49.999985, -49.99998, -49.99966, -49.99999, -6.6602516, 0.7095566, -0.013660431]

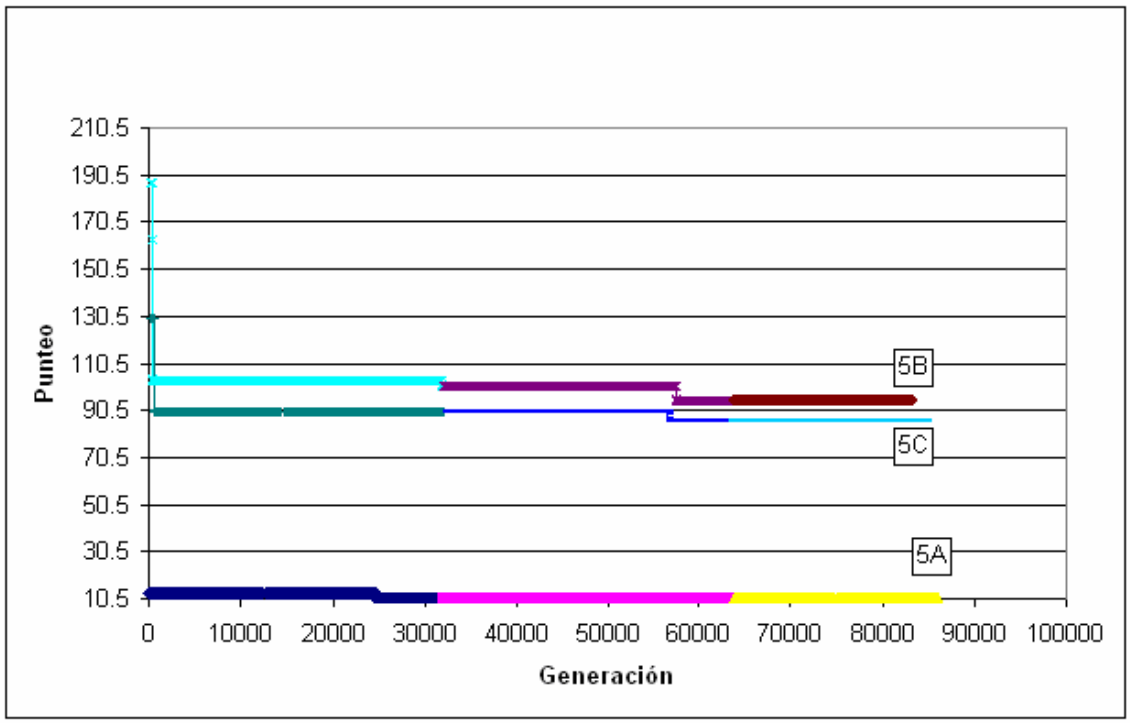
Y el mejor individuo del experimento fue:

[49.999985, 49.999954, -49.99997, -49.999985, -49.99999, -49.999992, -49.99999, -6.6298714, 0.7095566, -0.013660431]

Puede verse que el único factor que varió fue el segundo, lo cual indica que aproximadamente en la generación 24622 se alcanzó un punto de convergencia.

La figura 49 muestra una gráfica comparativa de los resultados obtenidos en los experimentos de este grupo, 5A, 5B y 5C. Puede observarse que en el experimento 5A se obtuvieron siempre los punteos menores, motivo por el cual fue el mejor experimento.

Figura 49. Comparación de los punteos obtenidos en el quinto grupo de experimentos, 5A, 5B y 5C



Además, tanto en el experimento 5B como en el 5C se observa el mismo comportamiento a largo plazo, se encuentra un punto de convergencia mucho antes de llegar al final de la ejecución.

Para el experimento 5B el último cambio significativo se alcanza en la generación número 57530 donde se pasa de un punteo de 101.012278 a 95.3095758.

Para el experimento 5C el último cambio significativo ocurre en las generaciones 56500 a la 56503 donde se pasa de un punteo de 89.4084509 a 87.0283822 y luego a 86.2136187. Puede concluirse entonces que la convergencia se alcanzó en estos experimentos mucho antes de llegar al final de la ejecución del algoritmo.

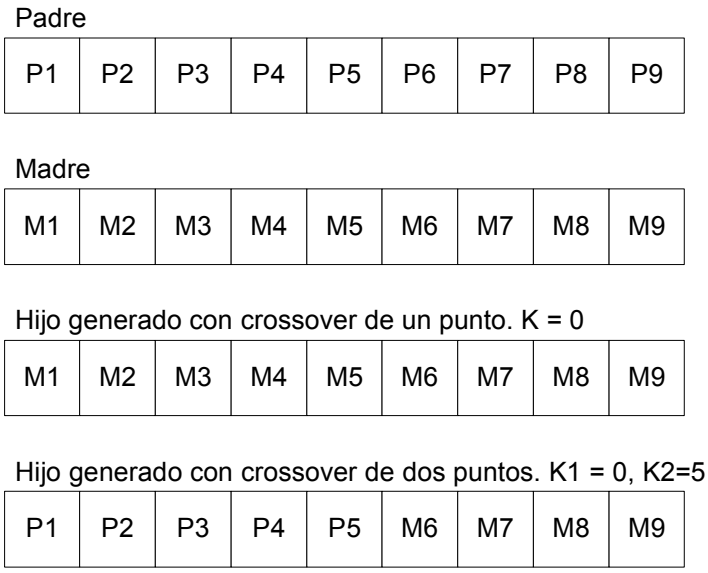
5.4. Tercer batería de experimentos

5.4.1. Cambios al algoritmo genético

En esta batería de experimentos se han realizado algunos cambios en la aplicación, que fueron determinados luego de varias pruebas:

- Se cambió el procedimiento que genera los factores numéricos para que redondee los valores, obteniendo así sólo números enteros.
- Se modificaron las funciones que realizan las operaciones de *crossover* de uno y dos puntos debido a que el primer punto de cruce a veces era cero, entonces para el *crossover* de un punto el individuo generado era copia del padre o de la madre, y el *crossover* de dos puntos se comportaba como un *crossover* de un punto, un ejemplo de este problema se muestra en la figura 50. Puede verse que para el *crossover* de un punto, cuando el punto de ruptura K=0 el hijo es una copia de la madre. Para el *crossover* de dos puntos, con los puntos de ruptura $K1$=0 y $K2$=5 el proceso de reproducción se comporta como crossover de un punto pues sólo existe un punto de ruptura.

Figura 50. Problemas en el *crossover* de uno y dos puntos cuando un punto de ruptura es cero



5.4.2. Conjunto de datos

En esta batería de experimentos se empleó el mismo conjunto de datos utilizado en la primera batería (ver sección 5.2.1), donde todos los periodos de tiempo son iguales a cien.

Se empleó nuevamente este conjunto de datos debido a que es más útil al momento de evaluar los resultados del algoritmo genético, pues se conoce de antemano cuál es el modelo matemático que representa los datos:

$$Y = 100$$

Para un grado de polinomio igual a diez, el individuo que se ajusta perfectamente a los datos es:

[100, 0, 0, 0, 0, 0, 0, 0, 0, 0]

5.4.3. Diseño del experimento

Para estos experimentos primero se realizaron varias ejecuciones de validación de parámetros, variando los parámetros en cada ejecución. Luego de varias ejecuciones se escogió la siguiente configuración de parámetros:

- Grado del polinomio: 10
- Factor numérico mínimo: -100
- Factor numérico máximo: 100
- Tamaño de la población: 10000
- Tipo de crossover: Dos puntos
- Mutaciones: 5000
- Mutar al mejor individuo: no
- Tipo de función de evaluación: PEMA (Porcentaje de Error Medio Absoluto)
- Divisor para la función de evaluación: 1
- Tipo de selección: Elitista
- Número de individuos mejores a elegir: 5000
- Condición de finalización: 3000 ejecuciones.

Es importante notar los siguientes cambios en la configuración de parámetros utilizada, con respecto a los experimentos de las baterías anteriores:

- Se incrementó considerablemente el tamaño de la población. Anteriormente había sido de 1000 individuos y ahora se incrementó a 10000.

- El divisor de la función de evaluación es uno, lo cual indica que no se altera el resultado de la función de evaluación original PEMA. En estos experimentos los resultados obtenidos representarán el porcentaje de error real del modelo matemático.

Se ejecutarán tres experimentos con la misma configuración de parámetros, como se muestra en la tabla XII.

Tabla XII. Diseño de la tercer batería de experimentos

	Experimento		
Grupo	A	B	C
1	Experimento 6ª Parámetros 1	Experimento 6B Parámetros 1	Experimento 6C Parámetros 1

5.4.4. Resultados

Los primeros dos experimentos de este grupo, 6A y 6B se ejecutaron 3000 veces pues se realizaron sin supervisión, pero en el tercer experimento, 6C, se detuvo la ejecución poco después de haber alcanzado el punto óptimo.

Experimento 6A

- Tiempo de ejecución: 1 hora y 51 minutos
- Número de ejecuciones: 3000
- Mejor punteo de la última ejecución: 0.0
- Mejor individuo: [100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
- Mejor generación: 445
    - Punteo: 0.0
    - [100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

Experimento 6B

- Tiempo de ejecución: 2 horas y 20 minutos
- Número de ejecuciones: 3000
- Mejor punteo de la última ejecución: 0.0
- Mejor individuo: [100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
- Mejor generación: 151
    - Punteo: 0.0
    - [100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

Experimento 6C

- Tiempo de ejecución: 17 minutos
- Número de ejecuciones: 338
- Mejor punteo de la última ejecución: 0.0
- Mejor individuo: [100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
- Mejor generación: 215
    - Punteo: 0.0
    - [100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

Puede observarse en este grupo de experimentos que todos encontraron el punto óptimo, donde el error del modelo es igual a cero, es decir el modelo matemático se ajusta perfectamente a los datos del historial.

El modelo matemático encontrado por el algoritmo genético en los tres experimentos es:

$$Y = 100 \cdot X^0 + 0 \cdot X^1 + 0 \cdot X^2 + 0 \cdot X^3 + 0 \cdot X^4 + 0 \cdot X^5 + 0 \cdot X^6 + 0 \cdot X^7 + 0 \cdot X^8 + 0 \cdot X^9$$

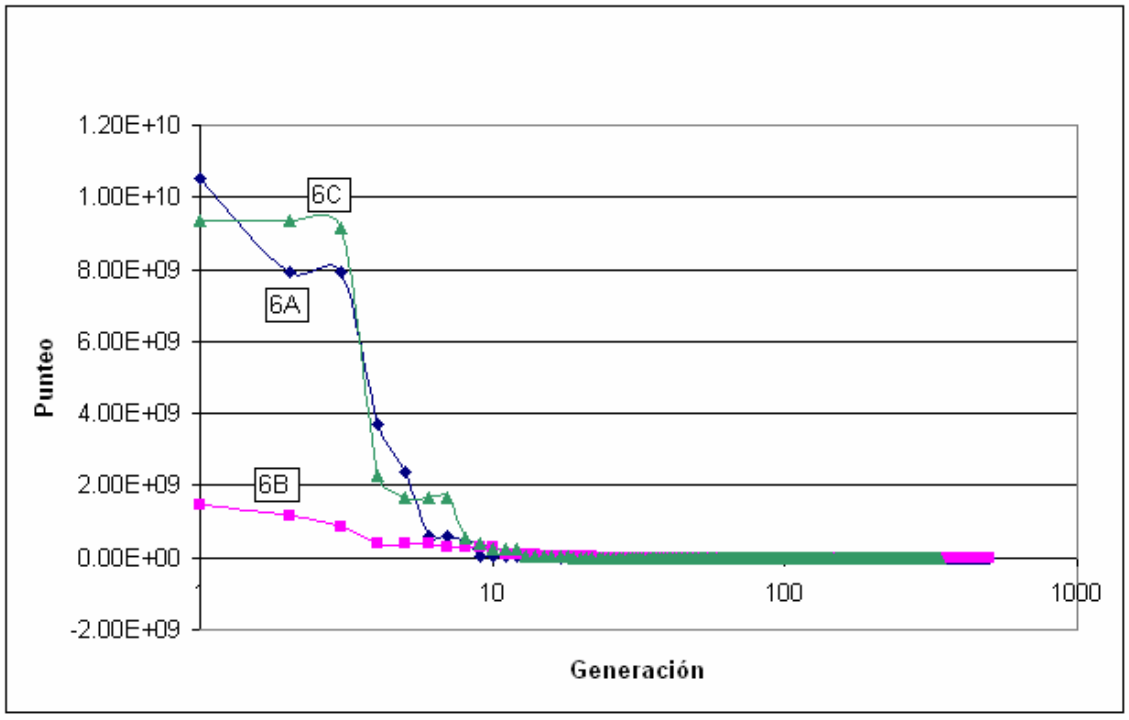
$$Y = 100$$

Dado el conjunto de datos utilizado es fácil verificar que el modelo matemático encontrado por el algoritmo genético es el óptimo, pues el historial de datos viene dado por una constante igual a cien.

La diferencia entre los experimentos de este grupo se encuentra en el tiempo que tardó cada uno en encontrar la solución óptima. El experimento 6B encontró primero una solución óptima, en la ejecución número 151.

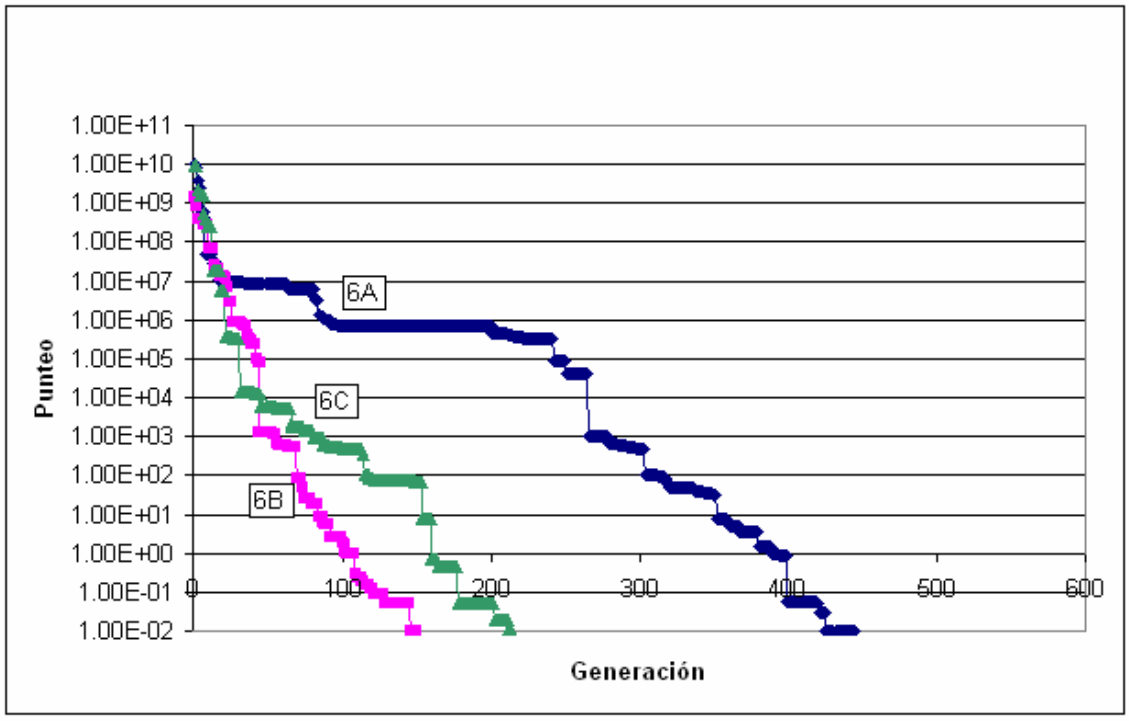
Una gráfica comparativa de los resultados obtenidos en este grupo de experimentos se muestra en la figura 51. Puede observarse que el experimento 6B inició con un mejor punteo que los otros experimentos.

Figura 51. Comparación de resultados obtenidos para el sexto grupo de experimentos, 6A, 6B y 6C



Si se modifica la escala del eje Y (punteo) a logarítmica, se obtiene la gráfica que se muestra en la figura 52. Aquí puede verse claramente que el experimento 6B llega primero al punto óptimo en cero, seguido del experimento 6C y por último el experimento 6A.

Figura 52. Comparación de resultados para el sexto experimento. Punteo con escala logarítmica



5.5. Cuarta batería de experimentos

5.5.1. Conjunto de datos y diseño

Debido a que en la tercera batería de experimentos el algoritmo genético encontró el modelo matemático óptimo para el conjunto de datos de prueba, se ha desarrollado una cuarta batera de experimentos que utilizará la misma configuración de parámetros pero para un conjunto de datos reales.

Los datos a analizar corresponden a las exportaciones de café de Guatemala durante treinta y seis meses, en el período de tiempo comprendido entre enero del 2001 y diciembre del 2003. Estos son los mismos datos que fueron utilizados en la segunda batería de experimentos y se muestran en la tabla X de la sección 5.3.1.

Los parámetros para esta batería de experimentos son:

- Grado del polinomio: 10
- Factor numérico mínimo: -100
- Factor numérico máximo: 100
- Tamaño de la población: 10000
- Tipo de *crossover*: Dos puntos
- Mutaciones: 5000
- Mutar al mejor individuo: no
- Tipo de función de evaluación: PEMA (Porcentaje de Error Medio Absoluto)
- Divisor para la función de evaluación: 1
- Tipo de selección: Elitista
- Número de individuos mejores a elegir: 5000
- Condición de finalización: 3000 ejecuciones.

Se ejecutarán nuevamente tres experimentos con los mismos parámetros, como se muestra en la tabla XIII.

Tabla XIII. Diseño de la cuarta batería de experimentos

	Experimento		
Grupo	A	B	C
1	Experimento 7A Parámetros 1	Experimento 7B Parámetros 1	Experimento 7C Parámetros 1

5.5.2. Resultados

Experimento 7A

- Tiempo de ejecución: 1 hora y 50 minutos
- Número de ejecuciones: 3000
- Mejor punteo de la última ejecución: 0.47442159972991
- Mejor individuo: [-32.0, 40.0, -1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
- Mejor generación: 234
    - Punteo: 0.47442159972991
    - [-32.0, 40.0, -1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

Experimento 7B

- Tiempo de ejecución: 1 hora y 52 minutos
- Número de ejecuciones: 3000
- Mejor punteo de la última ejecución: 0.47442159972991
- Mejor individuo: [-32.0, 40.0, -1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
- Mejor generación: 129
    - Punteo: 0.47442159972991
    - [-32.0, 40.0, -1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

Experimento 7C

- Tiempo de ejecución: 1 hora y 52 minutos
- Número de ejecuciones: 3000
- Mejor punteo de la última ejecución: 0.47442159972991
- Mejor individuo: [-32.0, 40.0, -1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
- Mejor generación: 128
    - Punteo: 0.47442159972991
    - [-32.0, 40.0, -1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

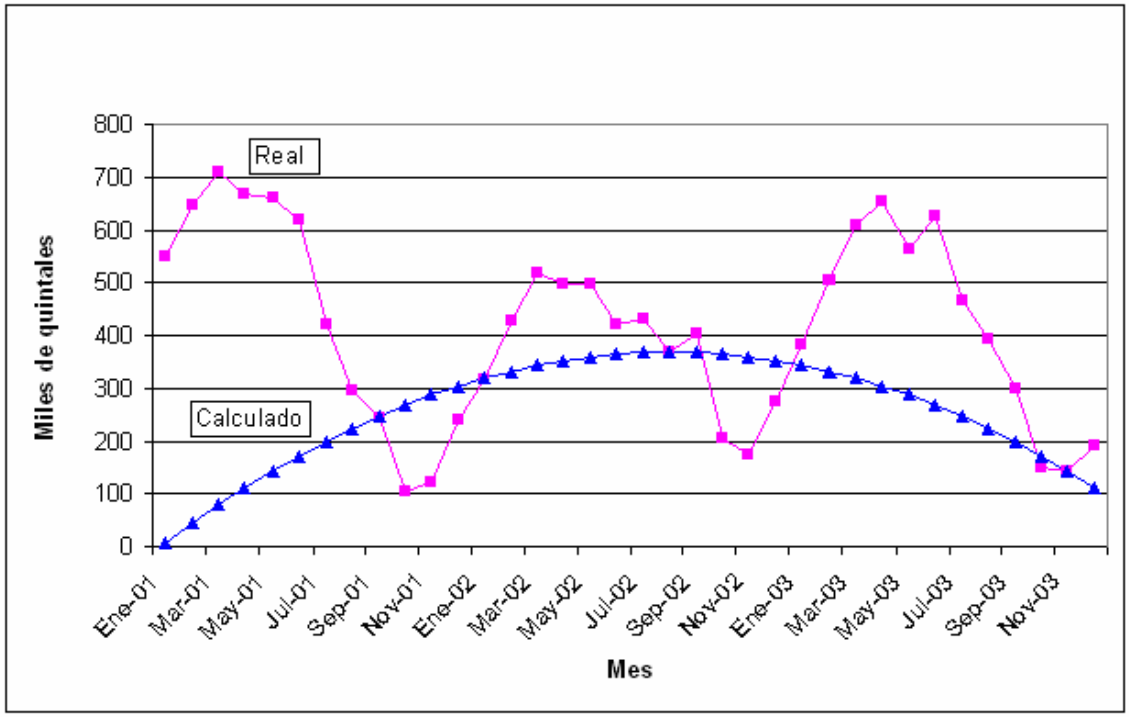
En este grupo de experimentos puede verse nuevamente que los tres experimentos convergen en el mismo punto, el cual presenta un porcentaje de error medio absoluto del 47%.

El modelo matemático encontrado por los tres experimentos fue:

$$Y = -32 + 40\,X - X^2$$

Una gráfica de los datos reales comparado con los datos obtenidos con el modelo se observa en la figura 53.

Figura 53. Comparación de las exportaciones de café reales contra las calculadas



Puede observarse que la gráfica de datos calculados no presenta un comportamiento parecido a los datos reales, además al final de la gráfica la tendencia de los datos calculados va hacia abajo mientras la tendencia de los datos reales va hacia arriba.

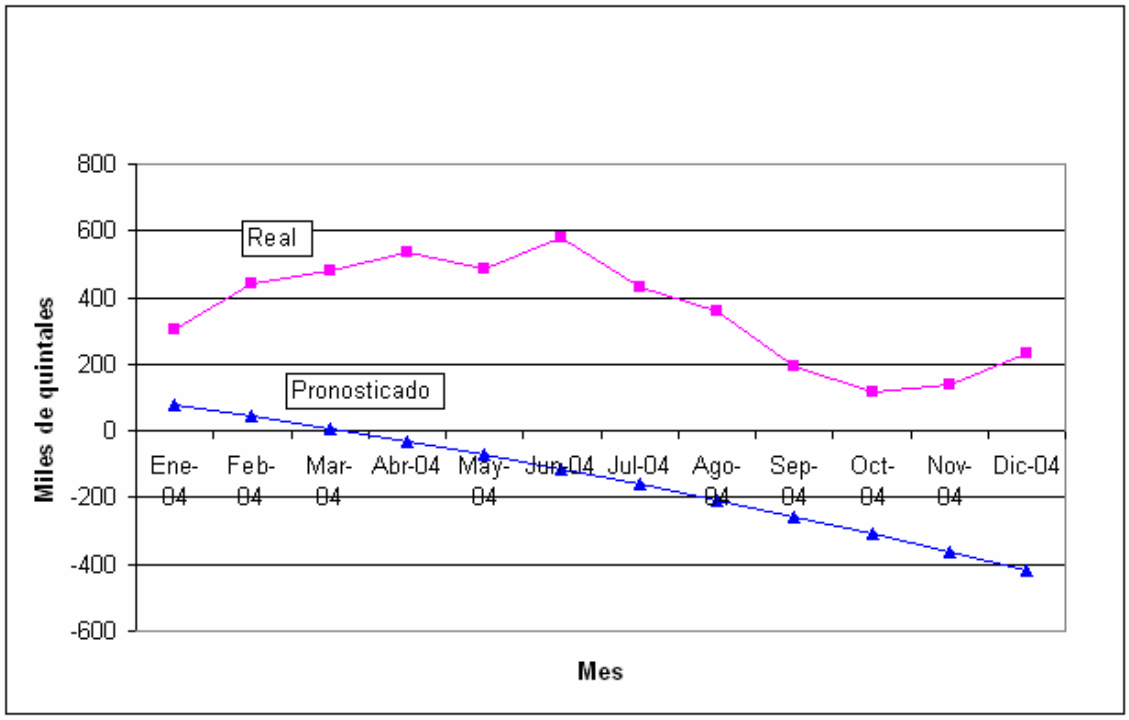
Ahora se pronosticarán los valores para las exportaciones de café del año 2004 utilizando el modelo matemático encontrado y se compararán con las exportaciones reales de ese año, las cuales son conocidas. Los resultados se muestran en la tabla XIV.

Tabla XIV. Datos reales de exportaciones de café y datos pronosticados con el modelo hallado en el experimento 7

Mes	Período	Real (miles de quintales)	Pronosticado (miles de quintales)
Ene-04	37	304.2	79
Feb-04	38	441.6	44
Mar-04	39	481.5	7
Abr-04	40	537.5	-32
May-04	41	485.5	-73
Jun-04	42	581.3	-116
Jul-04	43	430.6	-161
Ago-04	44	359.8	-208
Sep-04	45	196	-257
Oct-04	46	118.2	-308
Nov-04	47	136.2	-361
Dic-04	48	230.7	-416

La figura 54 muestra una gráfica comparativa entre los datos pronosticados con el modelo matemático y los datos reales de exportaciones de café para el período de tiempo comprendido entre enero y diciembre del año 2004.

Figura 54. Comparación de las exportaciones de café reales contra las pronosticadas



Puede observarse que la tendencia del modelo matemático va en dirección contraria a la de los datos reales. Este problema se ha presentado debido a que el comportamiento del historial de datos es cíclico, entonces al final de cada año los valores llevan una tendencia hacia abajo y al inicio del año llevan una tendencia hacia arriba.

Para resolver problemas de este tipo, donde los datos presentan un comportamiento cíclico, puede optarse por elaborar un modelo matemático para un ciclo que en el caso de las exportaciones de café sería un año, y luego utilizar ese modelo para pronosticar los datos de cada año. Si existe alguna tendencia además de los ciclos, hacia arriba o hacia abajo, puede además agregarse un incremento en los valores pronosticados que compense el valor de la tendencia.

Esto implica un análisis de los datos históricos que se poseen antes de desarrollar el modelo matemático, para determinar si existen ciclos o tendencias, éste análisis puede realizarse observando las gráficas del historial de datos. Luego del análisis puede decidirse que datos se emplearan para realizar el modelo matemático.

Los tres modelos matemáticos encontrados en esta batería de experimentos, 7A, 7B y 7C fueron iguales, por lo cual no es necesario realizar una comparación entre ellos para ver cuál es mejor. Cuando se tienen varios modelos matemáticos distintos es necesario evaluar cuál es mejor para pronosticar, para ello se puede utilizar cualquiera de las fórmulas de medición del error y elegir aquel modelo que presente un error más bajo.

Es recomendable utilizar el porcentaje medio de error (PME) en la comparación de modelos matemáticos, debido a que éste mide el porcentaje de error con signo, indicando así si el pronóstico es sobreestimado o subestimado.

En resumen, cuando se han obtenido varios modelos matemáticos con el algoritmo genético se debe seguir el siguiente proceso para seleccionar uno:

- Con cada modelo se debe realizar un pronóstico de los datos para un período de tiempo conocido. Como en el caso de las exportaciones de café para Guatemala, que se pronosticaron los datos para el año 2004, los cuales ya eran conocidos.
- Medir el error del pronóstico, comparando los datos reales contra los datos pronosticados mediante la función de porcentaje medio de error (PME).
- Elegir el modelo que presente un PME más cercano a cero.

5.6. Comportamiento de los parámetros del algoritmo genético

En esta sección se han comparado distintos conjuntos de parámetros del algoritmo genético y se presenta un resumen de los resultados obtenidos. Para estos experimentos se empleó un conjunto de datos hipotético para el cual se conocía el modelo matemático exacto que es un polinomio de grado 2, se seleccionó un polinomio de grado 2 debido a dos razones:

- Presenta cierto grado de dificultad lo que permite evaluar correctamente un conjunto de parámetros. Si los datos no presentan ninguna dificultad existe la probabilidad de que tanto un mal conjunto de parámetros como uno bueno encuentren el resultado óptimo.
- Se realizaron pruebas con polinomios de grado mayor pero no se llegaba a la solución óptima donde el error es cero, entonces se decidió tomar el conjunto de datos donde el algoritmo genético si hallaba la solución óptima global.

El conjunto de datos empleado se muestra en la tabla XV. Estos datos son representados exactamente por el siguiente modelo matemático:

$$Y = X^2 + X + 10$$

Así, el individuo óptimo a buscar es:

[10.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

Tabla XV. Datos de prueba para evaluar comportamiento de parámetros del algoritmo genético

Período	Valor
1	12
2	16
3	22
4	30
5	40
6	52
7	66
8	82
9	100
10	120
11	142
12	166
13	192
14	220
15	250
16	282
17	316
18	352
19	390
20	430

Se realizaron experimentos para evaluar el comportamiento de:

- Tipo de selección y *crossover*
- Número de mutaciones
- Tamaño de la población

5.6.1. Tipo de selección y *crossover*

Primero se compararon diferentes tipos de selección y crossover manteniendo constantes los demás parámetros. El conjunto de parámetros empleado fue:

- Grado del polinomio: 10
- Factor numérico mínimo: -200
- Factor numérico máximo: 200
- Tamaño de la población: 10000
- Tipo de *crossover*: Un punto, dos puntos, uniforme y de dos puntos asimétrico.
- Mutaciones: 100 (1%)
- Mutar al mejor individuo: no
- Tipo de función de evaluación: PEMA (Porcentaje de Error Medio Absoluto)
- Divisor para la función de evaluación: 1
- Tipo de selección: Elitista, generacional y hacinamiento determinista.
- Número de individuos mejores a elegir: 5000
- Condición de finalización: 3000 ejecuciones o cuando el error es cero.

Los resultados comparados fueron:

- Tiempo de ejecución en minutos
- Mejor generación
- Número de ejecuciones
- Error

Los resultados obtenidos en esta primera comparación se resumen en la tabla XVI.

Tabla XVI. Resultados para distintos tipos de selección y *crossover*

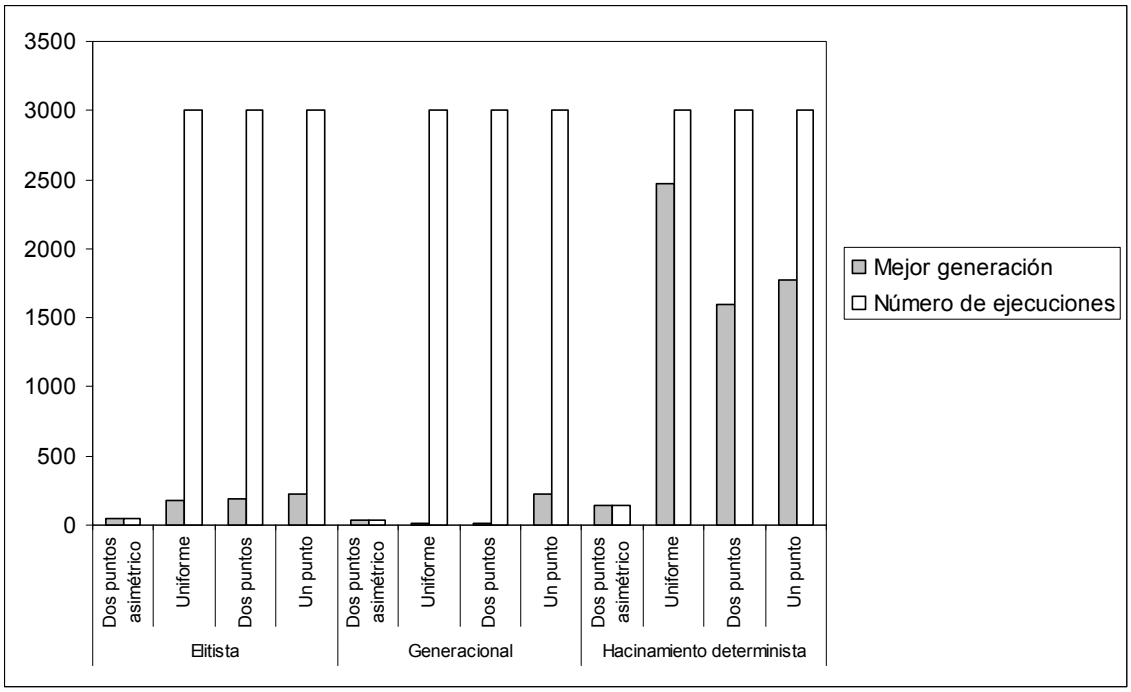
Tipo selección	Tipo *crossover*	Tiempo (minutos)	Mejor generación	Número de ejecuciones	Error
Elitista	Dos puntos asimétrico	1.13	50	50	0.00
	Uniforme	78.05	172	3000	800628.39
	Dos puntos	78.02	184	3000	392862.95
	Un punto	61.02	223	3000	1470758.74
Generacional	Dos puntos asimétrico	0.78	33	33	0.00
	Uniforme	67.10	10	3000	1009448.65
	Dos puntos	65.15	16	3000	569072.25
	Un punto	63.93	224	3000	2260162.58
Hacinamiento determinista	Dos puntos asimétrico	7.80	139	139	0.00
	Uniforme	189.85	2473	3000	113161.54
	Dos puntos	222.93	1598	3000	18862.79
	Un punto	170.88	3000	299055.255	1770

En general puede observarse que se obtienen mejores resultados al utilizar *crossover* de dos puntos asimétrico para los tres tipos de selección empleados. Además, sólo con el *crossover* de dos puntos asimétrico se logró encontrar la solución óptima donde el error es igual a cero, esto se debe a que éste método de cruce introduce mayor diversidad en la población evitando que el algoritmo converja en un óptimo local.

Al comparar la columna de la mejor generación con el número de ejecuciones en los casos donde se llegó al máximo de ejecuciones (3000) puede verse que al emplear los tipos de selección elitista y generacional el algoritmo encontró al mejor individuo al inicio de su ejecución y no mejoró con el tiempo, sin embargo con hacinamiento determinista los individuos mejoraron en etapas más avanzadas de la ejecución.

Esto se debe a que el hacinamiento determinista introduce mayor diversidad y explora un área mayor de búsqueda pues no selecciona una gran cantidad de individuos aptos como sucede en la selección elitista y generacional, sino que compara parejas aleatorias de individuos por lo que individuos menos aptos al inicio tienen mayor probabilidad de supervivencia y se mantiene así mayor diversidad en cada generación. Una gráfica de la mejor generación comparada con el número de ejecuciones total se muestra en la figura 55.

Figura 55. Mejor generación y número total de ejecuciones por tipo de selección y *crossover*



Con respecto al tiempo, la ejecución del algoritmo cuando se empleó hacinamiento determinista fue más lenta debido a que este método de selección realiza más comparaciones y operaciones sobre los individuos. Esto puede observarse gráficamente en la figura 56.

Figura 56. Tiempos por tipo de selección y *crossover*

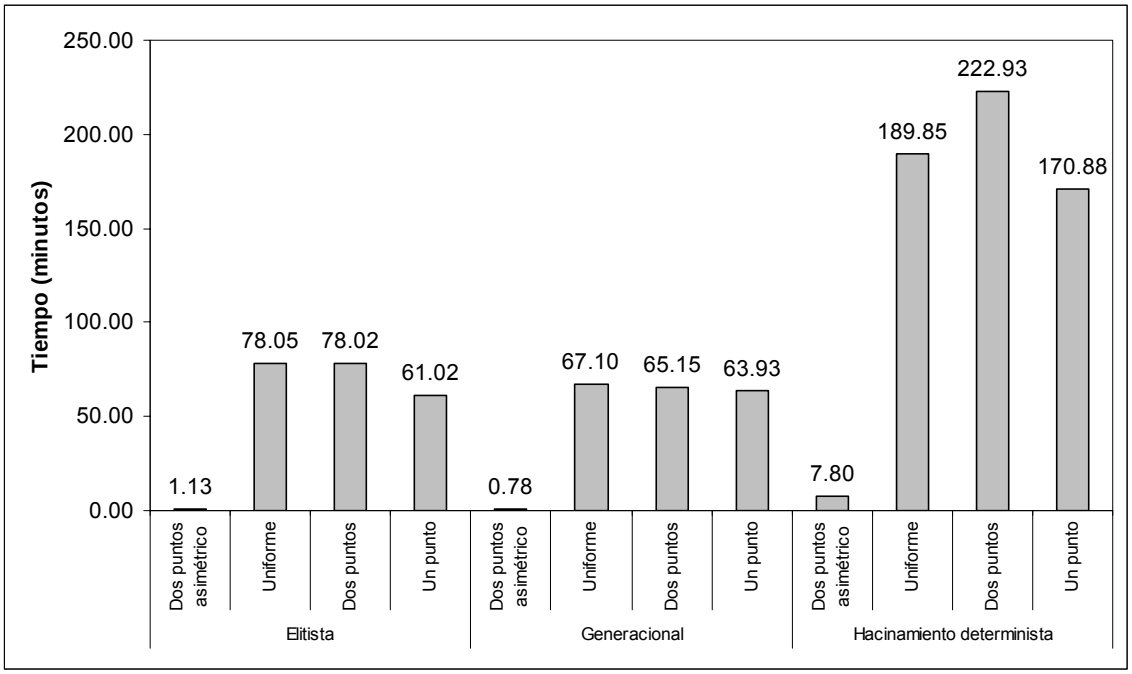
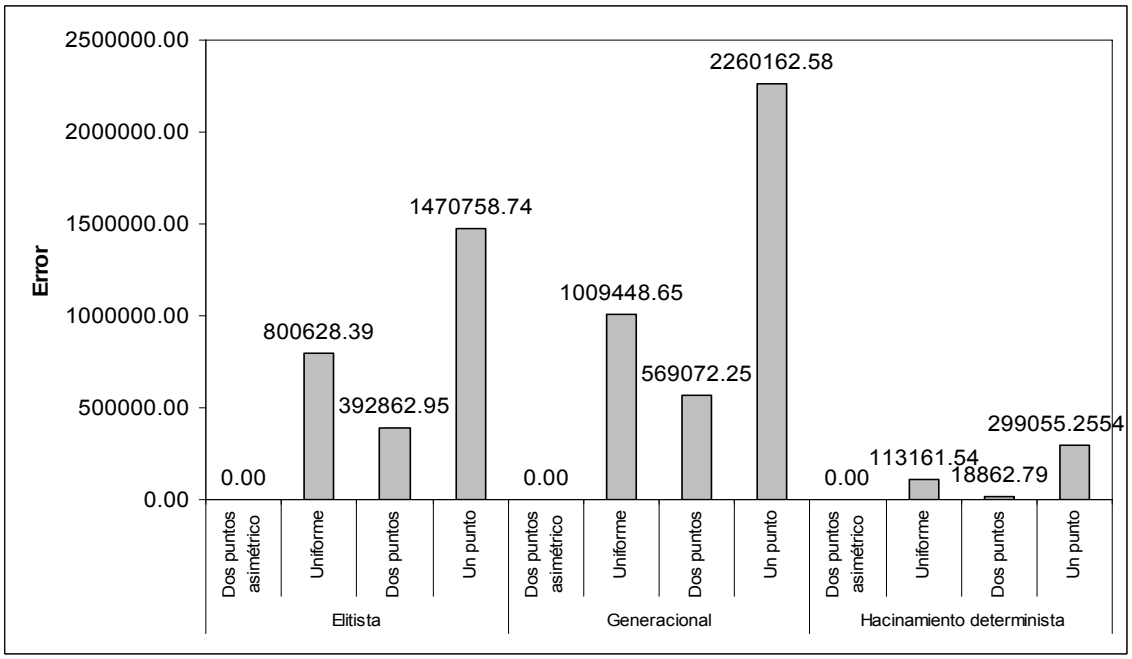


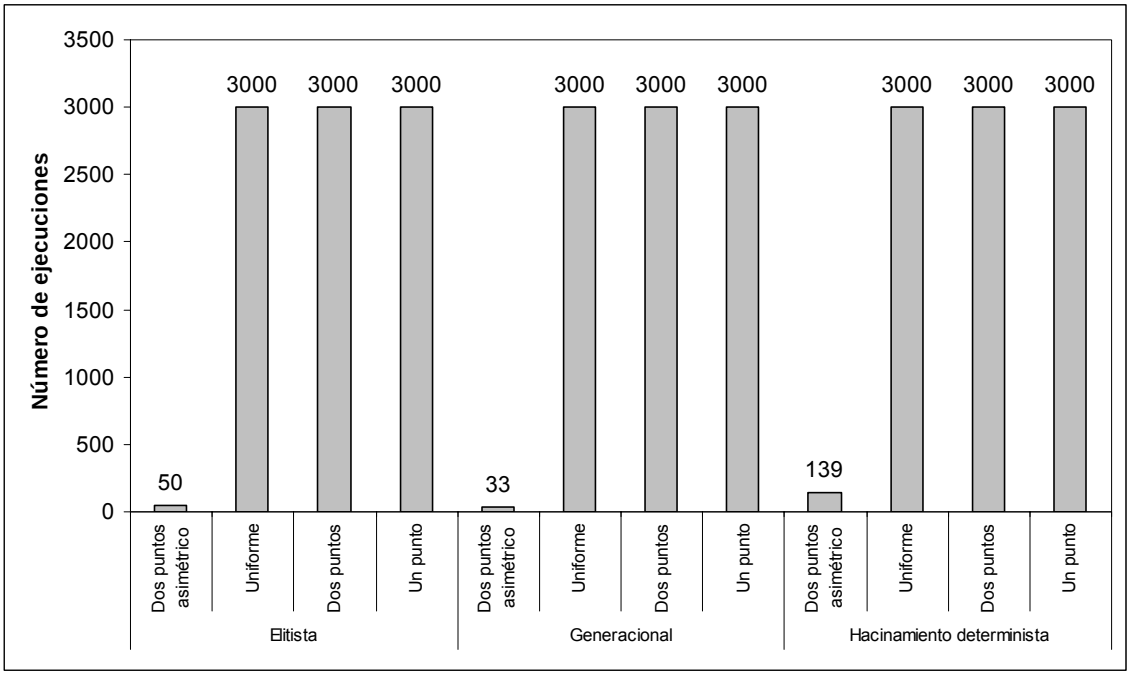
Figura 57. Error por tipo de selección y *crossover*



Una gráfica de los errores obtenidos para las distintas combinaciones de parámetros se presenta en la figura 57. Puede observarse que los mejores resultados se obtuvieron al emplear hacinamiento determinista.

Una gráfica del número de ejecuciones se muestra en la figura 58. Se puede observar que el número de ejecuciones cuando se empleó *crossover* de dos puntos asimétrico fue menor para la selección generacional (33) comparado con las ejecuciones que se realizaron con selección elitista (50) y hacinamiento determinista (139). Esto indica que el algoritmo tuvo una convergencia más rápida al usar selección generacional, de hecho si se observa nuevamente la figura 55 puede verse que la selección generacional y elitista converge más rápido que el hacinamiento determinista pues se estancaron rápidamente en un óptimo local aunque existían mejores individuos que sólo fueron explorados por el hacinamiento determinista.

Figura 58. Número de ejecuciones por tipo de selección y *crossover*



5.6.2. Mutaciones

Para evaluar el comportamiento del número de mutaciones, se empleó el siguiente conjunto de parámetros:

- Grado del polinomio: 10
- Factor numérico mínimo: -200
- Factor numérico máximo: 200
- Tamaño de la población: 10000
- Tipo de *crossover*: De dos puntos asimétrico.
- Mutaciones: Variable
- Mutar al mejor individuo: no
- Tipo de función de evaluación: PEMA (Porcentaje de Error Medio Absoluto)
- Divisor para la función de evaluación: 1
- Tipo de selección: Generacional.
- Número de individuos mejores a elegir: 5000
- Condición de finalización: 3000 ejecuciones o cuando el error es cero.

El número de mutaciones se fue variando desde 100 (1%) hasta 10000 (100%), los resultados obtenidos se resumen en la tabla XVII.

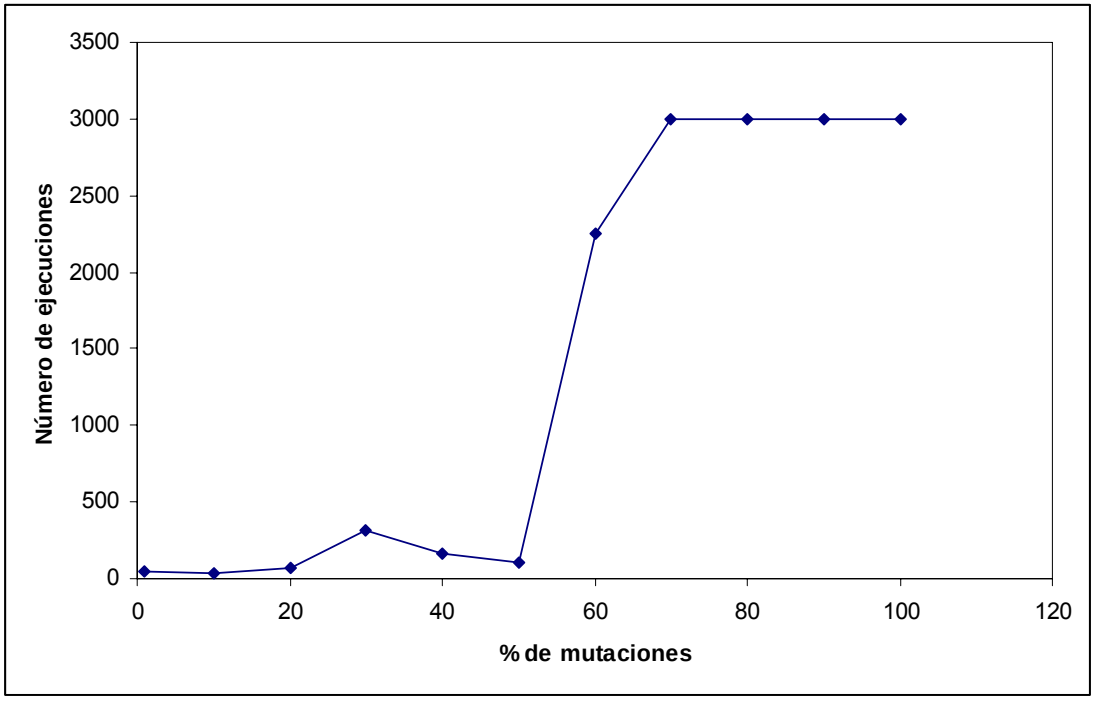
Puede observarse que conforme se va incrementando el porcentaje de mutaciones en la población se requieren de más ejecuciones del algoritmo para encontrar la solución y al utilizar 70% o más de mutaciones ya no se encontró la solución óptima al llegar a las 3000 ejecuciones que fueron configuradas.

Tabla XVII. Resultados para distintos porcentajes de mutación

Número de mutaciones	Porcentaje	Tiempo (minutos)	Número de ejecuciones	Error
100	1	1.016666667	41	0
1000	10	0.9	36	0
2000	20	1.75	67	0
3000	30	8.71666667	312	0
4000	40	4.3	158	0
5000	50	2.28333333	107	0
6000	60	60.7	2250	0
7000	70	87.6333333	3000	0.03557933
8000	80	87.4333333	3000	0.03557933
9000	90	99.75	3000	0.05421614
10000	100	82.7166667	3000	0.05555526

El número de ejecuciones requeridas para los distintos porcentajes de mutaciones se muestra en la figura 59. Puede observarse en esta gráfica que sólo existe una ligera mejoría en la ejecución del algoritmo al pasar de 1% a 10% de mutaciones, luego de este punto fue necesario realizar cada vez un mayor número de ejecuciones.

Figura 59. Número de ejecuciones para distintos porcentajes de mutaciones

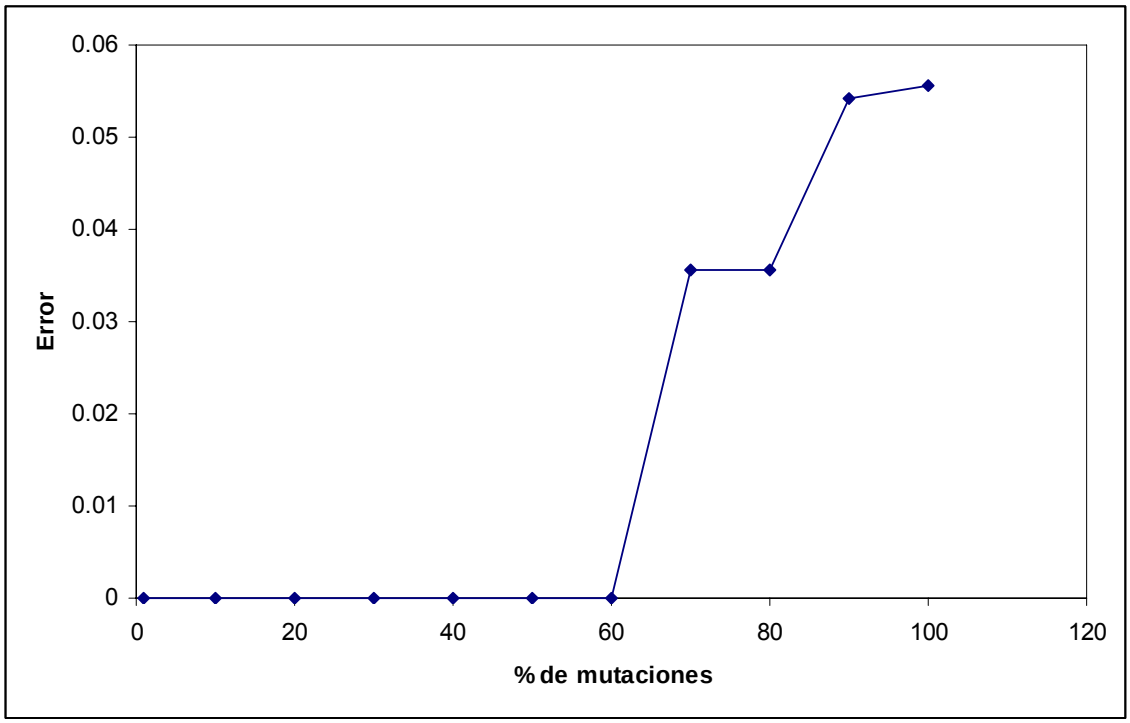


Las mutaciones se emplean debido a que introducen diversidad en la población, sin embargo al introducir demasiadas mutaciones la población se vuelve excesivamente diversa y el área de búsqueda del algoritmo se vuelve demasiado extensa por lo que tarda más en converger. Al emplear *crossover* asimétrico de dos puntos se ha introducido la diversidad suficiente en el algoritmo para que explore el área donde se encuentra la solución óptima por ello no se ve una mejora al incrementar el porcentaje de mutaciones.

Por otra parte, al emplear *crossover* de un punto, dos puntos y uniforme, no se introduce suficiente diversidad en la población y si resulta útil emplear un gran porcentaje de mutaciones para poder explorar el área donde se encuentra la solución óptima, tal y como ocurrió en la tercera batería de experimentos vista en la sección 5.4 donde se empleó un 50% de mutaciones y *crossover* de dos puntos.

En la figura 60 se muestran los errores obtenidos para los distintos porcentajes de mutaciones. Puede observarse que luego de emplear 70% de mutaciones ya no se encontró al individuo óptimo al finalizar las 3000 ejecuciones que estaban configuradas para el algoritmo. Además, al incrementar el porcentaje de mutaciones luego del 70% el error del mejor individuo encontrado en cada experimento fue incrementándose. Esto, al igual que el número de ejecuciones, indica que al incrementar las mutaciones la aptitud de los individuos fue empeorando. El uso excesivo de mutaciones introdujo diversidad en la población pero también destruyó a aquellos individuos que tenían buena aptitud.

Figura 60. Error obtenido para distintos porcentajes de mutaciones



5.6.3. Tamaño de la población

Para evaluar el comportamiento del algoritmo con distintos tamaños de población, se empleó el siguiente conjunto de parámetros:

- Grado del polinomio: 10
- Factor numérico mínimo: -200
- Factor numérico máximo: 200
- Tamaño de la población: Variable
- Tipo de *crossover*: De dos puntos asimétrico.
- Mutaciones: 1% (varía dependiendo el tamaño de la población)
- Mutar al mejor individuo: no
- Tipo de función de evaluación: PEMA (Porcentaje de Error Medio Absoluto)

- Divisor para la función de evaluación: 1
- Tipo de selección: Generacional.
- Número de individuos mejores a elegir: 50% (varía dependiendo el tamaño de la población)
- Condición de finalización: 3000 ejecuciones o cuando el error es cero.

Los resultados obtenidos se resumen en la tabla XVIII.

Tabla XVIII. Resultados para distintos tamaños de población

Tamaño de la población	Tiempo (minutos)	Número de ejecuciones	Error
100	8.58333333	3000	5.32E+07
1000	15.7666667	2570	0
2000	1.88333333	468	0
3000	0.36666667	61	0
4000	0.8	90	0
5000	0.55	50	0
6000	0.61666667	45	0
7000	0.81666667	42	0
8000	0.86666667	42	0
9000	0.9	38	0
10000	1.3	49	0
15000	1.88333333	40	0
20000	2.91666667	43	0
30000	5.78333333	41	0

Únicamente al emplear una población de 100 individuos no se encontró la solución óptima (error igual a cero), cuando se emplearon tamaños de población mayores o iguales a 1000 individuos siempre se halló la solución óptima.

La figura 61 muestra el número de ejecuciones requerido para los distintos tamaños de población. Puede observarse un cambio significativo en el número de ejecuciones requerido para encontrar la solución óptima al incrementar la población desde 100 hasta 3000 individuos, al emplear poblaciones mayores a 3000 individuos los cambios ya no fueron tan drásticos.

Figura 61. Número de ejecuciones para distintos tamaños de población

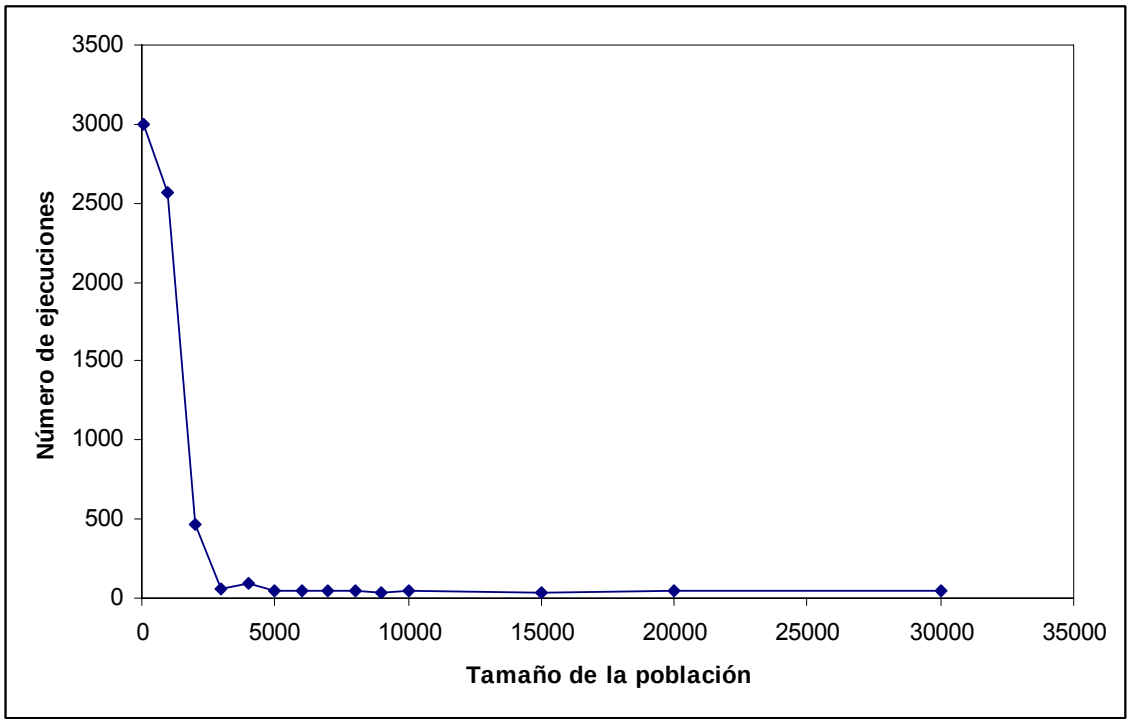
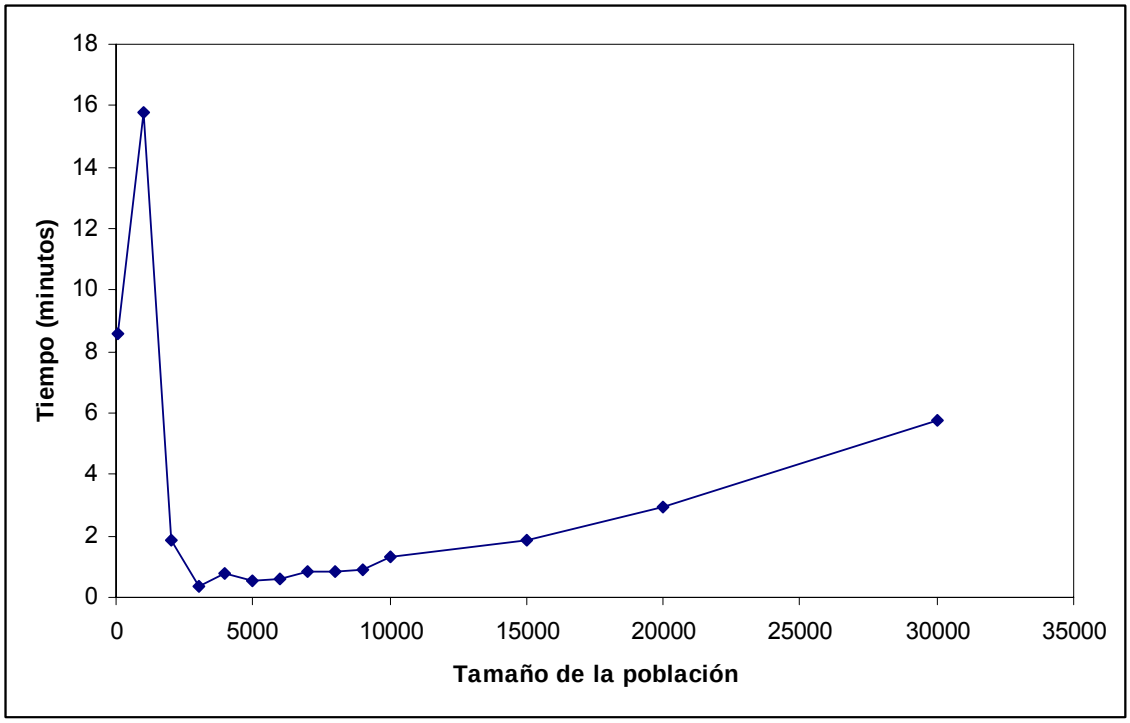


Figura 62. Tiempo de ejecución para distintos tamaños de población



Si se observan los tiempos de ejecución requeridos para cada tamaño de población (figura 62) puede verse igualmente una disminución significativa en el tiempo requerido hasta cuando se utilizó una población de 3000 individuos, de allí en adelante el tiempo de ejecución fue incrementándose.

Al incrementar el tamaño de la población existe mayor probabilidad de encontrar al individuo óptimo, sin embargo llega un punto donde el incremento del tamaño de la población ya no mejora la probabilidad de hallar a un individuo más apto y sólo se incrementa el tiempo de ejecución del algoritmo porque el área de búsqueda es mayor. Entonces debe emplearse un tamaño de población que brinde un área de búsqueda lo suficientemente grande pero que no incremente demasiado el tiempo de ejecución.

5.7. Resultados para distintos grados de polinomio

De acuerdo a las pruebas por parámetro efectuadas en la sección anterior, se determinó que el algoritmo genera mejores modelos cuando se emplea *crossover* asimétrico de dos puntos. Para un polinomio de grado dos, sin importar el método de selección empleado, el algoritmo siempre encontró el modelo exacto (ver tabla XVI).

Al comparar los métodos de selección se encontró que el mejor era el de hacinamiento determinista, pues introduce mayor diversidad en la población hallando así mejores soluciones. Al observar nuevamente la figura 57 puede verse que el hacinamiento determinista dio errores menores a los otros métodos de selección.

En esta sección se presentan los resultados obtenidos para distintos grados de polinomios empleando *crossover* asimétrico de dos puntos y hacinamiento determinista.

Los resultados obtenidos para polinomios de grado 0, 1, 2, 3 y 4 se resumen en la tabla XIX.

Tabla XIX. Resultados para polinomios de distintos grados

Grado	Ecuación	Error	Número de ejecuciones
0	$Y = 100$	0	106
1	$Y = 10X$	0	164
2	$Y = X^2 + X + 10$	0	139
3	$Y = X^3 - 27X^2 + 175X + 250$	0.01868 (1.868%)	3000
4	$Y = -0.6X^4 + 15X^3 - 11X^2 + 391X + 25$	0.13482 (13.482%)	3000

Puede observarse en esta tabla que se encontró el modelo exacto (error igual a cero) para los polinomios de grado 0, 1 y 2. Para los polinomios de grado 3 y 4 no se encontró el modelo exacto luego de 3000 ejecuciones del algoritmo genético.

Ahora se procederá a probar el algoritmo con datos reales, para ello se emplearán los datos históricos de las exportaciones de café de Guatemala para el año 2004 mostrados en la tabla XX. Se ha tomado solo un año debido a que las exportaciones de café de Guatemala presentan un comportamiento cíclico que se repite cada año y al incluir más de un ciclo existe mayor dificultad para hallar el modelo matemático con el algoritmo genético, como sucedió en la cuarta batería de experimentos (ver sección 5.5.2).

Tabla XX. Exportaciones de café de Guatemala para el año 2004

Mes	Miles de quintales
Enero	304.2
Febrero	441.6
Marzo	481.5
Abril	537.5
Mayo	485.5
Junio	581.3
Julio	430.6
Agosto	359.8
Septiembre	196
Octubre	118.2
Noviembre	136.2
Diciembre	230.7

Fuente: Banco de Guatemala. www.banguat.gob.gt

Para estos datos se obtuvieron los siguientes resultados:

- Error: 0.21399 (21.399 %)
- Número de ejecuciones: 3000
- Tiempo de ejecución: 2 horas 15 minutos y 16 segundos
- Modelo: 193.0 + 177.0 X - 28.0 X^2 + X^3

La gráfica de los datos reales comparada con la gráfica del modelo encontrado se muestra en la figura 63. Puede observarse que el modelo se ajusta al comportamiento de los datos reales, sin embargo al finalizar la gráfica la tendencia del modelo es hacia abajo mientras la tendencia de los datos reales va hacia arriba.

El modelo por tanto no es útil para pronosticar, porque para los siguientes periodos de tiempo va disminuyendo el valor dando incluso valores negativos, mientras los datos reales van en aumento. Esto puede observarse en la figura 64 donde se han pronosticado los periodos 13 al 16.

Figura 63. Gráfica real versus gráfica del modelo de pronóstico

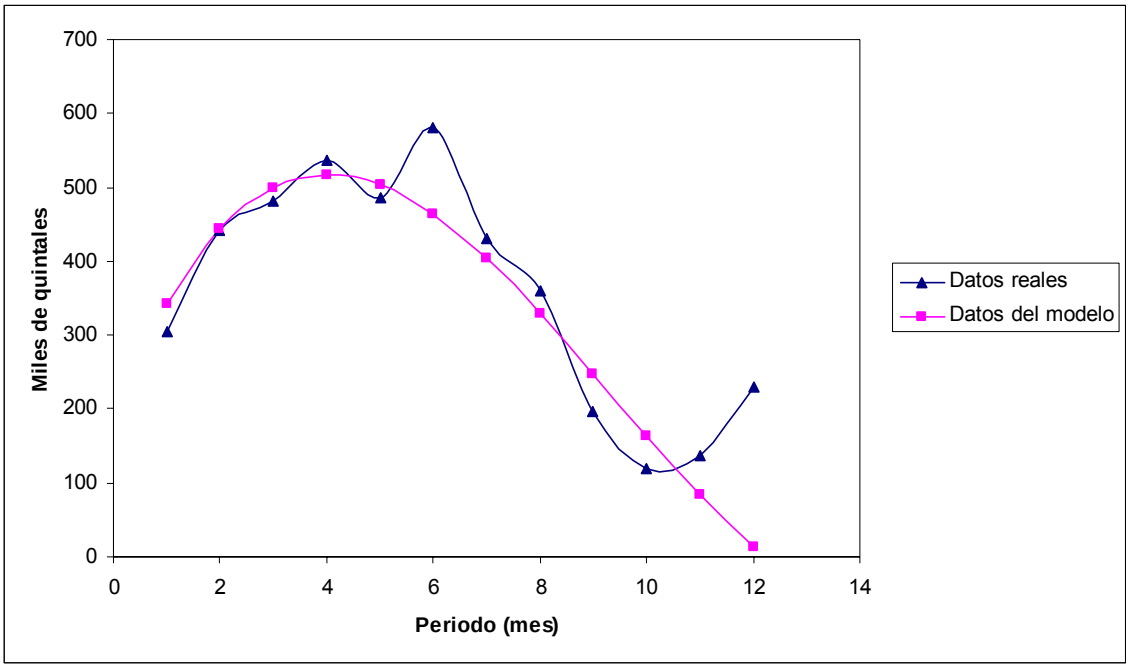
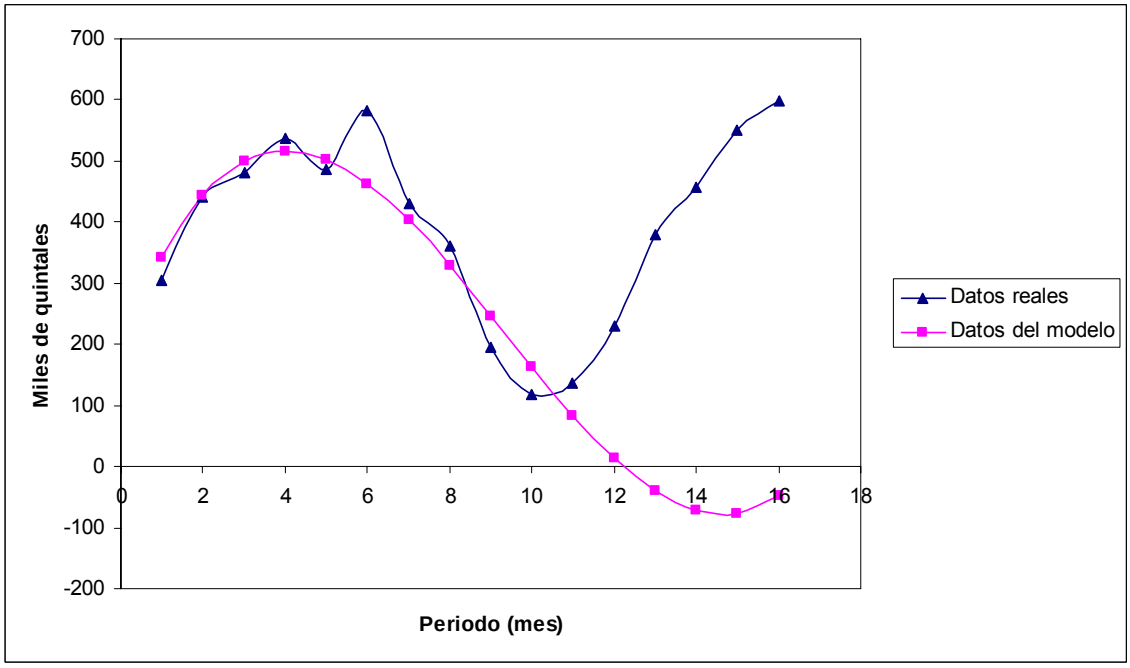


Figura 64. Pronóstico de periodos 13 al 16 con el modelo



5.8. Trabajo futuro

Hasta el momento se ha utilizado un modelo matemático en forma de polinomio para realizar los pronósticos, sin embargo los datos en estudio no siempre presentan un comportamiento que se pueda representar con un polinomio, existen casos donde los datos pueden ser representados de mejor forma con otra función, por ejemplo exponencial o logarítmica.

Para realizar un mejor modelo de pronóstico debe realizarse un análisis previo de los datos y determinar cuál es la función que se adapta mejor a su comportamiento y luego utilizar esa función como modelo de pronóstico.

Otra línea de trabajo futuro se encuentra en la codificación utilizada para los individuos de la población, aquí se trataron dos tipos de codificación una con números reales y otra con números enteros y se encontró que era mejor la codificación con números enteros porque reducía el espacio de búsqueda.

La codificación mediante números binarios reduce aún más el espacio de búsqueda y se adapta mejor a la forma en que trabajan los algoritmos genéticos, sin embargo hay que buscar una forma de codificación que permita una aplicación adecuada de los operadores genéticos.

Para los pronósticos interesa que al aplicar un operador genético el nuevo individuo mantenga las mejores características de los originales, esto implica que no puede utilizase una codificación binaria simple traduciendo directamente los valores binarios a decimales, pues al variar un bit de un número binario el cambio en el número decimal es completamente distinto. Una buena codificación binaria para este caso puede utilizar los códigos de gray, en los cuales el cambio de un bit del número binario representa un número decimal contiguo.

Finalmente, pueden implementarse otros procesos de selección y otros operadores genéticos que mejoren la diversidad del algoritmo genético tal como selección por torneo y selección por rueda de ruleta.

CONCLUSIONES

1. Los pronósticos intentan predecir el valor de una variable en el tiempo, su análisis puede realizarse cualitativamente, basándose en opiniones de expertos cuando no existe mucha información sobre el comportamiento pasado de la variable, o cuantitativamente mediante un análisis numérico de un historial de datos de la variable. La mejor forma de pronosticar es utilizar una combinación de ambos métodos, utilizando un modelo numérico apoyado en el juicio de una persona que domine el área en estudio.

2. Los algoritmos genéticos buscan soluciones a problemas mediante técnicas que imitan el proceso de reproducción natural, tal como selección, cruce y mutación. Por su flexibilidad pueden ser utilizados para resolver una gran variedad de problemas, aunque su uso principal es para buscar máximos o mínimos de funciones matemáticas. Su gran flexibilidad tiene como debilidad que no son especialistas en ningún área específica.

3. Los algoritmos genéticos no pueden utilizarse, directamente, para el pronóstico de una variable debito a su carácter estocástico, pues en dos ejecuciones diferentes se obtendrán siempre dos resultados diferentes. Para solventar este problema se diseño un algoritmo genético que encuentre un modelo matemático dado por un polinomio que se utiliza para pronosticar, luego, los valores de una variable.

4. El aproximar los datos del historial de una variable a una función no siempre es adecuado, debido a que existe más de una función con el mismo comportamiento en un intervalo de valores, pero al ampliar este intervalo sólo una de ellas, o ninguna, pronosticará valores futuros de la variable en forma adecuada, esto debe evaluarse, cuidadosamente, mediante el juicio del pronosticador.

   La codificación de los individuos del algoritmo genético es importante debido a que ésta define el espacio de búsqueda. En el algoritmo genético desarrollado se encontraron mejores resultados al utilizar una codificación de individuos con números enteros que cuando se emplearon números reales, debido a que los números enteros reducían el espacio de búsqueda para el algoritmo.

5. La diversidad de la población es importante para que el algoritmo genético pueda encontrar una buena solución, esta diversidad puede ser introducida mediante operadores genéticos como cruce y mutación ó mediante métodos de selección, para el algoritmo genético empleado se introdujo diversidad en la población empleando mutaciones *crossover* asimétrico de dos puntos y selección mediante hacinamiento determinista.

   Se obtuvieron mejores resultados con el *crossover* asimétrico de dos puntos que cuando se empleó *crossover* de un punto, dos puntos y uniforme, también se obtuvieron mejores resultados cuando se empleó el método de selección por hacinamiento determinista que con la selección elitista y generacional.

Al emplear *crossover* asimétrico de dos puntos y selección generacional se obtuvieron mejores resultados cuando se introdujo 1% de mutaciones, al emplear un mayor porcentaje la población tardó más tiempo en converger pues se alteraban individuos que presentaban buenas características.

El tamaño de la población fue un parámetro clave para el algoritmo genético desarrollado. En una población grande la probabilidad de encontrar un buen individuo se incrementa, así en los experimentos cuando se empleó una población grande se obtuvieron resultados con un error igual a cero. El incrementar el tamaño de la población ocasionó un incremento en el tiempo necesario para generar una nueva población, sin embargo, también, se redujo el número de ejecuciones necesarias para encontrar al individuo deseado y el tiempo total de ejecución del algoritmo fue menor que cuando se utilizó una población más pequeña. Sin embargo, llega un punto donde al incrementar más el tamaño de la población sólo incrementa el tiempo de ejecución y no la probabilidad de encontrar mejores individuos, este punto se determinó mediante experimentos donde se variaba el tamaño de la población manteniendo los demás parámetros constantes.

6. Con el algoritmo genético desarrollado se encontró el modelo matemático exacto en la tercer batería de experimentos cuando los datos del historial eran constantes, sin embargo cuando se utilizaron datos variables y cíclicos en la cuarta batería de experimentos el mejor modelo encontrado presentaba un error de 47% y no era adecuado para pronosticar.

Al experimentar con polinomios de diferente grado se encontró que el algoritmo genético hallaba un modelo de pronóstico óptimo, donde el error era igual a cero, únicamente para polinomios de grado cero, uno y dos, para polinomios de mayor grado ya no se encontró el modelo exacto. Aún al experimentar con un solo ciclo de datos para disminuir la variabilidad de la entrada el algoritmo genético encontró un modelo de pronóstico con un error de 21%, el cual tampoco era adecuado para pronosticar porque seguía una tendencia contraria a la real. Entonces, el algoritmo genético no resulto útil para hallar modelos matemáticos de pronóstico con forma de un polinomio cuando los polinomios eran de grado dos o superior. En los experimentos realizados sólo resulto útil para datos hipotéticos, pero no para datos reales.

RECOMENDACIONES

1. Al diseñar un algoritmo genético debe utilizarse una codificación de los individuos que tenga un espacio de búsqueda finito y reducido, de ser posible debe utilizarse una codificación binaria.

2. Se debe efectuar un análisis previo de los datos históricos de la variable en estudio antes de buscar un modelo matemático con el algoritmo genético desarrollado, éste análisis debe enfocarse en determinar tendencias y comportamiento cíclico en los datos. Luego, debe decidirse qué datos se emplearán como entrada en el algoritmo genético para encontrar un modelo matemático adecuado. Si los datos históricos presentan ciclos es aconsejable utilizar sólo los datos de un ciclo como entrada para el algoritmo genético.

3. El modelo matemático obtenido con el algoritmo genético debe someterse al juicio del pronosticador ó de alguien experto en al área para la que desee utilizarse, si es aprobado entonces podrá utilizarse para pronosticar.

4. Se deben realizar siempre varias ejecuciones del algoritmo genético con los mismos parámetros ya que debido a su comportamiento estocástico pueden obtenerse distintos modelos en cada ejecución. Luego, se debe evaluar mediante una función de medición de distancias el error del pronóstico para determinar cuál es el mejor modelo, acompañando siempre esta decisión del juicio del pronosticador.

ANEXOS

A.1. Ejecución de la aplicación

A.1.1. Pantalla principal

Para ejecutar la aplicación primero se debe cargar un archivo con el historial de datos eligiendo el menú archivo y luego cargar datos. Este archivo debe tener un valor por cada línea. El siguiente es un ejemplo de archivo con cinco valores históricos de una variable.

958
732
586
125
600

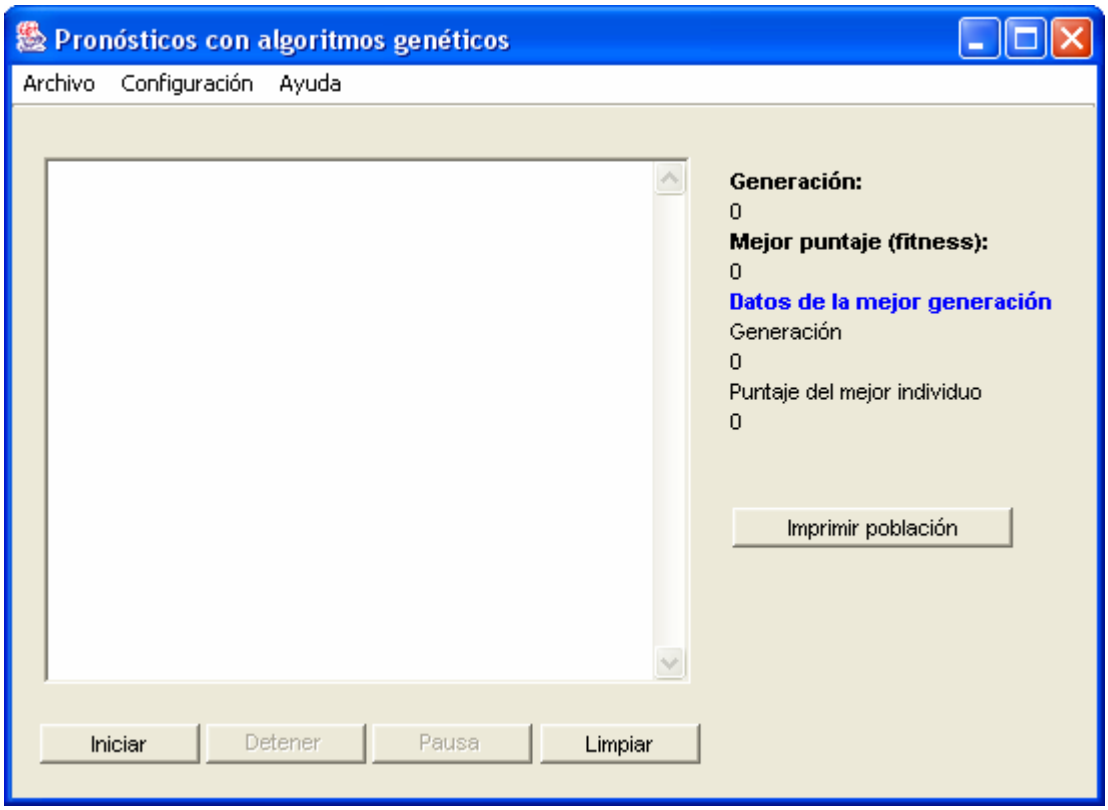
Cada línea representa un período de tiempo que puede ser año, mes, día, hora, etc.

La pantalla principal de la aplicación se muestra en la figura 65. En la pantalla principal aparecen cinco botones, con la siguiente función.

- Iniciar: inicia la ejecución del algoritmo genético.
- Detener: detiene la ejecución del algoritmo genético, reiniciando todos los valores de ejecución.

- Pausa: realiza una pausa en la ejecución, todos los valores se mantienen y al reanudar la ejecución ésta continúa a partir del último punto.
- Limpiar: limpia los resultados mostrados en el cuadro de texto.
- Imprimir población: escribe a un archivo los individuos de la población actual.

Figura 65. Pantalla principal de la aplicación



En el cuadro de texto se muestra la información del mejor individuo de cada generación y su punteo. Adicionalmente, se muestran dos etiquetas con el número de generación actual y el puntaje del mejor individuo encontrado en esa generación.

La información que se muestra en el cuadro de texto, también, se escribe a un archivo de bitácora con el siguiente formato:

Generación: *número de generación*
Mejor individuo: [factor 1, factor 2, factor 3,…, factor *n*]
Mejor punteo: *punteo del mejor individuo*

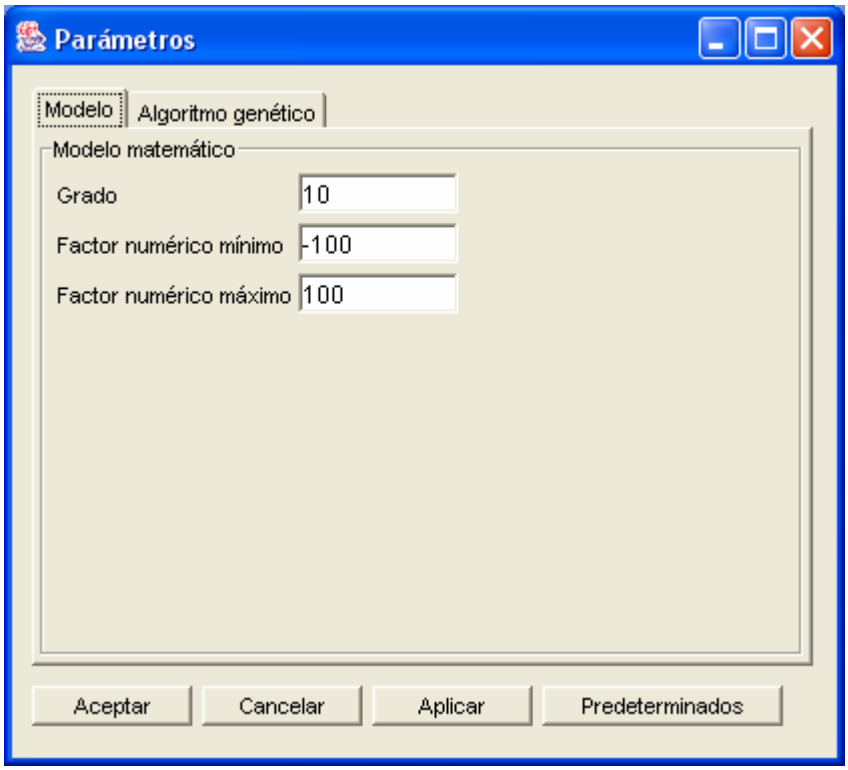
Una vez finalizada la ejecución del algoritmo genético, el modelo matemático a utilizar para pronosticar estará dado por el polinomio que se forme con los factores numéricos del mejor individuo.

A.1.2. Pantalla de parámetros

Para modificar los parámetros de la aplicación, se debe acceder al menú de configuración y luego elegir parámetros.

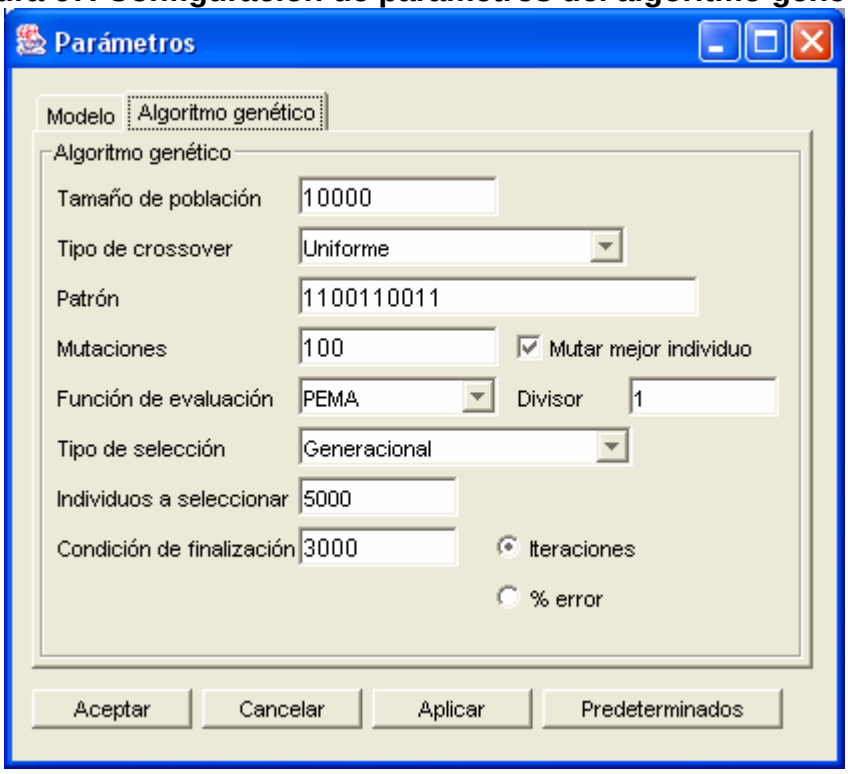
La ventana de configuración de parámetros tiene dos pestañas, en la primera se muestran los parámetros propios del modelo matemático que son el grado del polinomio y el rango dentro del cual se generarán los factores numéricos, esto se muestra en la figura 66.

Figura 66. Configuración de parámetros para modelo matemático



La segunda pestaña muestra los parámetros propios del algoritmo genético que incluyen el tamaño de la población, tipo de *crossover*, número de mutaciones, función de evaluación, tipo de selección y condición de finalización, como se puede observar en la figura 67. Para una explicación detallada de cada uno de ellos puede consultar la sección 4.2.1.

Figura 67. Configuración de parámetros del algoritmo genético



REFERENCIAS ELECTRÓNICAS

1. Campos Y., Rahn A. **Algoritmo evolutivo multi-objetivo para maximizar una función de dos variables.** Disponible: http://www.alfredorahn.com/docs/AG_Bivariado.pdf

2. Carlos A. Coello Coello. **Introducción a los algoritmos genéticos.** Disponible: http://www.redcientifica.com/doc/doc199904260011.html

3. Darrell Whitley. **A genetic algorithm tutorial.** Computer Science Department, Colorado State University. Marzo 1993. Disponible: http://www.science.uva.nl/~mes/psdocs/ga-tr103.ps.gz

4. Jorge Martín Martín, Diego García Morate. **Seminario: Algoritmos** genéticos**.** Universidad de Valladolid, España. Junio, 2005. Disponible: http://www.infor.uva.es/~calonso/IAI/TrabajoAlumnos/memoriaAG.pdf

5. Rafael Penaloza Nyssen. “Apología de los algoritmos genéticos”, **Laberintos e infinitos**, (8). 2004. Disponible: http://laberintos.itam.mx/PDF/num8/203

6. **The Genetic Algorithms Archive.** Diponible: http://www.aic.nrl.navy.mil/galist/

REFERENCIAS

1. Carlos Moreno. **Pronósticos: Modelo cualitativo de pronósticos y aplicaciones modelos de series de tiempo.** Facultad de Administración de Empresas, Universidad de La Salle. Bogotá, Colombia. Disponible: http://www.gestiopolis.com/recursos/documentos/fulldocs/ger1/serietiempo.htm

2. Render Barry, et. al. **Principios de Administración de Operaciones**. Editorial Prentice Hall Hispanoamericana. México, 1996.

3. Nojek S., Britos P., Rossi B. y García Martínez. **Pronóstico de ventas: Comparación de predicción basada en redes neuronales versus método estadístico.** Departamento de Ingeniería Industrial, Instituto Tecnológico de Buenos Aires. Argentina. Disponible: http://www.gacetafinanciera.com/PROY_EXCEL/pronosticodeventas.pdf

4. Adam Marczyk. **Algoritmos genéticos y computación evolutiva.** 2004. Disponible: http://the-geek.org/docs/algen/algen.html

5. Carmen Cortés Parejo. **Algoritmos genéticos y problemas de visibilidad.** Universidad de Sevilla. Junio de 1996, p.1. Disponible: http://ma1.eii.us.es/miembros/cobos/articulo/Tesinas/Ccortes.pdf

6. Eduardo F. Morales. **Búsqueda optimización y aprendizaje. Algoritmos genéticos.** Tecnológico de Monterrey. Noviembre 2004. Disponible: http://ccc.inaoep.mx/~emorales/Cursos/NvoAprend/node105.html

7. David Santo Orcero. **Los algoritmos genéticos.** Disponible: http://www.orcero.org/irbis/disertacion/node1.html

8. Juan Julián Merelo Guervós. **Informática evolutiva: Algoritmos genéticos.** Disponible: http://geneura.ugr.es/~jmerelo/ie/ags.htm

9. Javier Ezcaray, José Delgado. **Un paseo por la jungla de la** computación **evolutiva.** El Rinconcito Informático. Febrero de 2004. Disponible: http://www.elrinconcito.com/articulos/Genetico/Geneticos.htm

10. Bo Yuan. **Deterministic crowding, recombination and self-similarity.** Scool of Information Technology and Electrical Engineering, The University of Queensland, Australia. 2002, pp.1-2. Disponible: http://www.itee.uq.edu.au/~boyuan/Mypaper/CEC02_7028.pdf

11. R. K. Bock y W. Krischer. **The data analysis briefbook.** CERN, Geneva. Disponible: http://rkb.home.cern.ch/rkb/titleA.html

12. David Sanz Hernanz, David Valle Millán. **Algoritmos genéticos.** Departamento de Informática, Universidad de Valladolid, España. Junio, 2005, pp.21-23. Disponible: http://www.infor.uva.es/~calonso/IAI/TrabajoAlumnos/AlgoritmosGeneticosValleMillan.pdf

13. Wayne L. Winsaston, **Investigación de operaciones: Aplicaciones y algoritmos.** Editorial Iberoamérica, México.

14. Pedro Y. Piñero, Leticia Arco, María M. García, Liesner Acevedo. **Algoritmos genéticos en la construcción de funciones de pertenencia borrosas.** Universidad central “Marta Abreu” de Las Villas, Cuba, pp.3-4. Disponible: http://tornado.dia.fi.upm.es/caepia/numeros/18/pinyero.pdf

15. http://jdom.org/

16. Artificial **Life And Other Experiments.** Disponible: http://www.aridolan.com/

17. Banco de Guatemala. www.banguat.gob.gt