Generación de Modelos de Estimación utilizando Programación Genética Lineal

Roberto Sánchez[1], Javier Martínez[1] y Benjamín Barán[1,2]
[1] Universidad Católica "*Ntra. Sra. de la Asunción*"
[2] Universidad Nacional de Asunción - Paraguay

Resumen. El uso de reglas de decisión y técnicas de estimación es cada vez más frecuente para la toma de decisiones. En los últimos años se realizaron estudios donde se aplica la programación genética (PG) en la obtención de reglas para realizar predicciones. Una nueva rama en el área de los algoritmos evolutivos es la programación genética lineal (PGL) que evoluciona secuencias de instrucciones de un lenguaje de programación imperativo. Este trabajo propone la obtención de reglas de decisión para la estimación de series de tiempo utilizando PGL. Los resultados obtenidos para el pronóstico del Índice de Precios al Consumidor (IPC) y el precio de la soja por tonelada, muestran el potencial de esta nueva propuesta.

Palabras Claves: Programación Genética, Indicadores Económicos, Series de Tiempo, Pronósticos

1. INTRODUCCIÓN

En los últimos años se ha puesto mayor interés en la aplicación de técnicas de inteligencia artificial para el análisis de mercados financieros. Esto se debe a la mayor disponibilidad de recursos computacionales así como al rápido acceso a un enorme volumen de datos. En este contexto, la aplicación de computación evolutiva permitiría obtener reglas de decisión para la predicción de indicadores económicos. El uso de reglas de decisión y técnicas de análisis en el mercado financiero se ha convertido en una rutina y estudios académicos recientes así lo demuestran [1,2].

La implementación de la programación genética ayuda a la obtención de modelos de decisión en los mercados financieros [1]. Por ejemplo, ya se publicaron trabajos donde se aplica la programación genética al mercado de intercambio de divisas [3]. En otro estudio, se trata de predecir el índice bursátil *Dow Jones* [4]. La predicción de la volatilidad en los mercados financieros es otra área de gran interés donde se ha utilizado *PG* con resultados alentadores [5].

La economía, el gobierno y las empresas financieras utilizan y dependen de los pronósticos para el desarrollo de sus actividades. Los métodos actuales para la predicción de las series de tiempo están sujetos a errores debido al comportamiento cambiante de las variables y a la intervención humana para la recolección de datos. Así, resulta necesario tener métodos que puedan adaptarse a condiciones cambiantes posibilitando pronósticos más precisos de series de tiempo.

Este trabajo estudia, entre otras series temporales, al IPC del Paraguay y al precio de la soja en toneladas. El índice de precios al consumidor (IPC), también utilizado como índice de inflación, representa los precios de una canasta de productos y su variación mide típicamente la modificación en los costos de estos productos. Este índice nos informa sobre el poder adquisitivo de la gente [6].

Este trabajo propone aplicar una nueva técnica en el área de la programación genética, la *Programación Genética Lineal (*en adelante PGL*)*, que en lugar de usar los tradicionales árboles de la programación genética clásica, evoluciona secuencias de instrucciones de un Lenguaje de Programación Imperativo [7]. Aunque el tiempo es una variable continua, en este estudio utilizaremos mediciones discretas, correspondientes a periodos aproximadamente equidistantes, como el mes o el año [6]. Es importante notar que se podrían identificar dos objetivos en el análisis de series temporales. El primero sería explicar las variaciones de la serie temporal en el pasado, tratando de determinar si corresponden a un patrón de comportamiento. Si se consigue encontrar el patrón o modelo, se intentará predecir el comportamiento futuro de la serie, que es el segundo objetivo del análisis [8].

2. PROGRAMACIÓN GENÉTICA

Los algoritmos evolutivos (AE) tienen en común el hecho de simular la evolución de una población de soluciones codificadas (individuos) manipuladas por un conjunto de operadores y evaluados por alguna función de *fitness* [9]. Esta función de *fitness* es un parámetro de adaptabilidad que determina la calidad de una solución.

Los AE se diferencian en la manera en que se representan las soluciones y los operadores evolutivos utilizados. Un área relativamente nueva dentro de los algoritmos evolutivos es la programación genética, donde programas de computadora evolucionan para intentar resolver problemas previamente definidos. Así, en la programación genética se representan las soluciones como programas de computadora, a diferencia de otros AE en los que los cromosomas representan parámetros a optimizar. La programación genética se define como una evolución directa de programas con el propósito de aprender por inducción, independiente de la representación de estos programas [7].

Los operadores genéticos deben garantizar la formación de programas sintácticamente correctos. Además, se debería intentar que los programas sean también semánticamente correctos.

Koza utilizó árboles sintácticos de un lenguaje de programación funcional para representar individuos [9]. En esta representación, las funciones están ubicadas en los nodos interiores del árbol y las hojas son constantes o variables de entrada. A este tipo de representación se la conoce como programación genética basada en árboles (*tree-based genetic programming, TGP*) [7].

3. PROGRAMACIÓN GENÉTICA LINEAL

Como la definición de programación genética deja libre el tipo de representación de los programas, se han desarrollado muchas variantes de programación genética [10]. Una variante prometedora es la programación genética lineal, en la cual los programas son representados como secuencias de instrucciones de un lenguaje de programación imperativo. En esta variante, en lugar de utilizar las tradicionales estructuras en forma de árbol, se evolucionan secuencias de instrucciones de un Lenguaje de Programación Imperativo como C, C++, Java, etc. La forma de representar una instrucción sería:

$$\mathbf{r_0 = r_1 + r_5}$$

donde $\mathbf{r_i}$ representa un registro. A este tipo de representación se lo conoce como código de tres direcciones, dado que se utilizan tres direcciones para representar la instrucción: una para el registro destino ($\mathbf{r_0}$ en el ejemplo), y dos para los registros operandos. Conforme se discute en [5], la utilización de instrucciones basadas en registros proporciona las siguientes ventajas:

- Mayor velocidad de ejecución.
- Aprovechamiento del espacio de almacenamiento, reutilizando registros.
- Posibilidad de optimizar el código.
- Facilidad para realizar operaciones genéticas (cruzamiento, mutación, etc.)
- Posibilidad de obtener resultados (*outputs*) múltiples.

En la programación genética lineal, el espacio del fenotipo es el conjunto de las funciones matemáticas y el espacio de genotipo es el conjunto de los programas en una cierta representación, compuestos de los elementos del lenguaje de programación utilizado. Los programas en PGL son representados como un conjunto de instrucciones que ejecutados de forma secuencial nos proporcionan la solución al problema en cuestión. Para este trabajo, las instrucciones pueden ser de dos o tres instrucciones. Cada instrucción está compuesta por:

- Un registro destino (*output*) donde se almacena el resultado obtenido.
- Uno o dos registros que almacenan los valores con los que se opera.
- Una operación del conjunto de operaciones definidas.

Una representación típica de un programa obtenido con PGL, utilizando instrucciones de 2 y 3 registros, sería por ejemplo:

```
r[3]=r[8]-r[1]
r[6]=r[6]-r[17]
r[1]=r[6]+r[1]
r[2]=seno[7]
r[1]=r[2]+r[2]
r[0]=r[1]*r[1]
```

Todos los registros almacenan valores, pero existe una cantidad de registros reservados para las constantes numéricas y están protegidos contra escritura. Estas constantes son inicializadas al comienzo del programa. Cada individuo consta de una secuencia de instrucciones, donde cada instrucción es representada como una *4-tupla*, <op,i,j,k>, donde el signo "op" representa la operación a efectuarse, la letra "i" es el índice del registro donde se almacenará el resultado, y las letras "j", "k" son los índices de los registros donde se encuentran los operandos. Para una *4-tupla* definida como <+,0,2,9> la instrucción seria: **r[0]=r[2]+r[9]**.

El número total de registros, incluyendo las constantes, suele definirse al inicio del programa. Para la mayoría de los problemas, no se necesita un número muy grande de registros, por lo general se utilizan 256 como máximo. Por convención, si la salida (*output*) es uno solo, se utiliza **r[0]**. Las entradas se almacenarían así a partir del registro **r[1]**, dejando al final registros para variables intermedias [7].

La programación genética lineal puede generar un código con una parte que influye el resultado y otra que parecen innecesarias pues no influyen en los cálculos de los registros de salida. Al primer grupo se le llama código efectivo, mientras que al segundo se lo conoce como código no-efectivo o *introns*. Estos *introns* generalmente resultan útiles en la evolución por dos motivos, porque reduce el efecto que puede tener una variación en el código efectivo y porque el código no efectivo permite variaciones que permanecen neutrales en términos de *fitness* [7]. Por lo tanto, incluso si un programa ya no puede crecer en tamaño por restricciones impuestas *a priori*, todavía puede crecer su código efectivo.

4. PLANTEAMIENTO DEL PROBLEMA

En este estudio se obtienen reglas o patrones para la estimación de series temporales, basadas en sus valores históricos, aplicando la programación genética lineal. Si representamos nuestro modelo de estimaciones como una función objetivo

$$f : I^n \rightarrow O^m \qquad \begin{cases} n & \text{cardinalidad del vector de entrada} \\ m & \text{cardinalidad del vector de salida} \end{cases} \qquad \textbf{(1)}$$

entonces los individuos pueden ser tratados como modelos de predicción que aproximan a la función objetivo. En (1), I^n representa los datos de entrada (*Input*) en un espacio de dimensión n y O^m representa los datos de salida (*Output*). Para este trabajo n tomará un valor en el rango [1,6] y m será igual a 1, dado que solo interesa calcular una única salida: el próximo pronóstico de la serie de tiempo estudiada.

El proceso evolutivo busca así un programa (o individuo) que represente la mejor estimación, utilizando datos históricos de entrenamiento. En forma general, se busca que a partir de suficientes datos de entrenamiento, se generen predicciones a futuro para datos aun desconocidos al momento de la estimación.

Para este trabajo se utiliza un conjunto de 256 registros, descriptos a continuación:

Registros Operando: tienen datos que serán utilizados en el cálculo.
Registro Destino: almacena el resultado de una operación realizada sobre el/los registro/s operando/s.

En este trabajo se define una cantidad fija de registros constantes, los cuales solo pueden ser Registros Operando. En cambio los registros que no fueron definidos como registros constantes, pueden ser Registros Operando y/o Registros Destino. Existen además registros especiales de entrada y salida de datos:

Registros Input: donde se almacenan datos de entrada antes de evaluar un programa (**r[10]** a **r[15]** en este trabajo).
Registros Output: donde se leerán los resultados luego de la ejecución de un programa. **r[0]** para este trabajo.

Con PGL tenemos dos tipos de operaciones, las operaciones propiamente dichas (aritméticas, exponenciales, trigonométricas, booleanas) y las operaciones condicionales. En la tabla 1 se resumen las operaciones más utilizadas.

Tabla 1: Operaciones en Programación Genética Lineal.

Tipo de Operación	Notación General	Rango Input
Aritméticas	$r_i = r_j + r_k$ $r_i = r_j - r_k$ $r_i = r_j * r_k$ $r_i = r_j / r_k$	$r_i, r_j, r_k \in \Re$
Exponenciales	$r_i = r_j \wedge r_k$ $r_i = \log(r_j)$	$r_i, r_j, r_k \in \Re$
Trigonométricas	$r_i = \text{sen}(r_j)$ $r_i = \cos(r_j)$	$r_i, r_j \in \Re$
Booleanas	$r_i = r_j$ and r_k $r_i = r_j$ or r_k	$r_i, r_j, r_k \in$ ß
Condicionales	if($r_j <= r_k$) if($r_j >= r_k$)	$r_i, r_j \in \Re$

En la implementación realizada para este estudio, se utilizan las operaciones Aritméticas, Exponenciales y Trigonométricas. No se implementan las operaciones Booleanas, ni Condicionales.

Un elemento fundamental del proceso evolutivo es la comparación entre individuos utilizando una medida de adaptabilidad o *fitness.* Existen numerosas técnicas para el cálculo del *fitness* [11]. Para el presente trabajo, el *fitness* de un individuo está determinado por el error total cometido con los datos de entrenamiento, utilizando como métrica de error el Error Medio Absoluto (2). El *fitness* es inversamente proporcional al error de aproximación, medido en relación a una serie temporal de entrenamiento con *N* muestras.

$$e = \frac{\sum_{j=1}^{N} \left| x_j - p_j \right|}{N} \qquad \text{donde: } x_j \text{ es el valor real y } p_j \text{ el pronóstico para el instante } j \qquad \textbf{(2)}$$

5. MÉTODOS TRADICIONALES

Para evaluar los resultados de esta propuesta, se la comparara con pronósticos realizados utilizando 3 métodos lineales ampliamente conocidos [13].

1- Promedio Móvil: donde se realiza un promedio de los últimos *k* valores para estimar el valor S_{t+1} de una variable aleatoria *x* para el siguiente instante (t+1).

$$S_{t+1} = \frac{\sum_{i=t-k}^{t} x_i}{k} \qquad \text{donde:} \quad \begin{cases} x & \text{valor real} \\ k & \text{cantidad de valores} \end{cases} \qquad \textbf{(3)}$$

2- Suavizado Exponencial: que estima el próximo valor como una suma ponderada dando mayor peso a las observaciones más recientes, utilizando una constante de suavizado α que toma valores en el intervalo [0,1].

$$S_{t+1} = \alpha x_t + (1-\alpha) S_t \quad \text{donde:} \quad \begin{cases} S_t & \text{estimación para el periodo} \\ \alpha & \text{constante de suavizado} \end{cases} \qquad \textbf{(4)}$$

Si se extiende la fórmula del suavizado exponencial vemos que es una suma ponderada de todos los valores históricos de *x*.

$$S_{t+1} = \alpha x_t + \alpha(1-\alpha) x_{t-1} + \alpha(1-\alpha)^2 x_{t-2} + ... + \alpha(1-\alpha)^t x_0 \qquad \textbf{(5)}$$

3- Suavizado Exponencial con Tendencia: que utiliza el método de suavizado exponencial y lo adapta para incluir un factor de tendencia.

$$S_{t+1} = \alpha x_t + (1-\alpha) S_t + T_t \quad \text{donde:} \quad \begin{cases} S_t & \text{estimación para el perioto t} \\ T_t & \text{tendencia para el periodo t} \end{cases} \qquad \textbf{(6)}$$

donde el cálculo de la tendencia se realiza conforme a la siguiente fórmula:

$$T_t = \beta L_t + (1-\beta) T_{t-1} \quad \text{donde:} \begin{cases} L_t & \text{variable de tendencia para el periodo } t \\ \beta & \text{constante de suavizado} \end{cases} \qquad \textbf{(7)}$$

Nuevamente se utiliza una constante de suavizado β, con valores en [0,1]. Por su parte, el cálculo de la Tendencia se realiza conforme:

$$L_{t+1} = \alpha(x_t - x_{t-1}) + (1-\alpha)(S_t - S_{t-1}) \qquad \begin{cases} x & \text{valor real} \\ S_t & \text{estimación para el periodo } t \end{cases} \qquad \textbf{(8)}$$

6. COMPARACIÓN DE PGL CON MÉTODOS TRADICIONALES

Este trabajo reporta resultados experimentales con un conjunto de cuatro series temporales de prueba. Dos de las mismas son series de tiempo obtenidas a partir de datos históricos del Banco Central del Paraguay: el precio de la soja en Toneladas y el Índice de Precios al Consumidor (IPC) [14]. Con fines de validación matemática, se utilizaron además dos series artificiales de tiempo, la primera es la función *seno(x)* mientras que la segunda es una serie de tiempo artificial, propuesto en [12] de la siguiente forma:

$$f_x = \begin{cases} seno\ (x) + \sqrt{x} & desde \quad 1 \le x \le 20 \quad (\text{segmento } 1) \\ x^2 + 2 & desde \quad 21 \le x \le 40 \quad (\text{segmento } 2) \\ seno\ (x) - \sqrt{x} & desde \quad 41 \le x \le 60 \quad (\text{segmento } 3) \end{cases} \qquad \textbf{(9)}$$

La serie producida por (9) trata de emular un comportamiento casi-cíclico [12].

7. RESULTADOS EXPERIMENTALES

Para validar la propuesta de utilizar PGL para aproximar series temporales, se realizaron pruebas experimentales con diversas series de tiempo, como las 4 arriba presentadas. Para obtener los resultados experimentales que siguen, se utilizaron 80 valores históricos, 60 de los cuales fueron utilizados para entrenar a los individuos (o programas) y los 20 restantes sirvieron para validar los modelos encontrados.

La programación genética lineal, como otros algoritmos evolutivos, es sensible a varios parámetros, como los presentados en la Tabla 3, donde se resume los valores utilizados en las implementaciones del presente trabajo. En este trabajo la condición de parada es una cantidad máxima de generaciones. Como el algoritmo implementado es elitista, la mejor solución encontrada no se pierde, siendo así la seleccionada como modelo predictivo para la serie en cuestión.

En las tablas 4 y 5 se presentan los errores (definido en la ecuación 2) obtenidos al aproximar las series temporales estudiadas con los diferentes modelos de predicción, utilizando 1.000, 5.000 y 10.000 generaciones respectivamente. Como puede observarse, los errores utilizando modelos encontrados con PGL fueron consistentemente menores. Resulta interesante destacar que el error para el precio de la soja por tonelada es menor con el modelo obtenido con 5.000 generaciones que con el modelo de 10.000 generaciones, debido al *sobreentrenamiento* que tuvieron los datos con más generaciones en el periodo de entrenamiento, lo que disminuye su generalidad para otros intervalos como los del periodo de validación.

Tabla 3: Parámetros utilizados para realizar los experimentos.

Tamaño máximo de programa para los individuos de la población inicial.	20
Tamaño mínimo de programa para los individuos de la población inicial	1
Tamaño máximo de programa	200
Cantidad de registros	20
Cantidad de registros variables	10
Cantidad de operadores	9
Cantidad de individuos en la población	1.000
Cantidad de individuos por torneo	4
Cantidad de torneos por generación	10
Cantidad de datos para validación del modelo	20
Cantidad de datos de entrenamiento	60
Probabilidad de reproducción entre ganadores del torneo	100%
Probabilidad de inserción de una instrucción entre ganadores del torneo	50%
Probabilidad de mutar (efectivo) entre ganadores del torneo	95%

Tabla 4: Errores experimentales al utilizar PGL con 1.000, 5.000 y 10.000 generaciones.

	PGL 1.000 generaciones	**PGL 5.000 generaciones**	**PGL 10.000 generaciones**
Seno	0,01	0,0074	0,000097
IPC	0,32	0,29	0,28
SOJA	7,97	7,68	7,84
Serie Artificial	346,14	338,32	311,09

Tabla 5: Errores experimentales al utilizar modelos estadísticos.

	Suavizado Exp. c/ Tendencia	**Suavizado Exponencial**	**Promedio Móvil**
Seno	0,75	0,69	0,81
IPC	0,39	0,48	12,79
SOJA	8,48	8,72	56,32
Serie Artificial	489,17	448,86	1201,56

Los experimentos fueron realizados en un computador personal con procesador de 2.4 GHz y 512 Megabytes de memoria RAM. Los algoritmos fueron implementados en C#. En la tabla 6 se presentan los tiempos de ejecución.

Tabla 6: Tiempo requerido para encontrar modelos predictivos con PGL para distintos números de generaciones.

	1.000 Generaciones	**5.000 Generaciones**	**10.000 Generaciones**
PGL	12 seg.	1 min 05 seg.	3 min. 21 seg.

En las figuras 3 y 4 se pueden observar series de prueba junto a las series generadas por los modelos predictivos para el periodo de validación, calculadas con

10.000 generaciones evolutivas, con datos del periodo de entrenamiento. Por cuestiones de espacio, no se presentan resultados para otras series de tiempo. Para obtener estos resultados se utilizaron seis registros de entrada con: los 4 últimos valores de la serie temporal estudiada, la última predicción y un contador del número de la predicción.

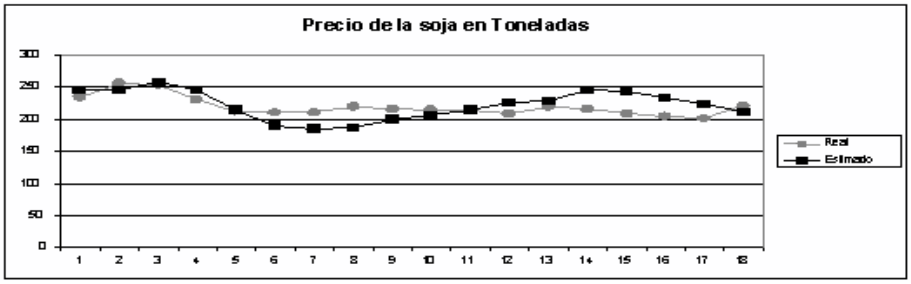

Fig. 3. Precio de la Soja y su estimación utilizando *PGL* con 10.000 generaciones.

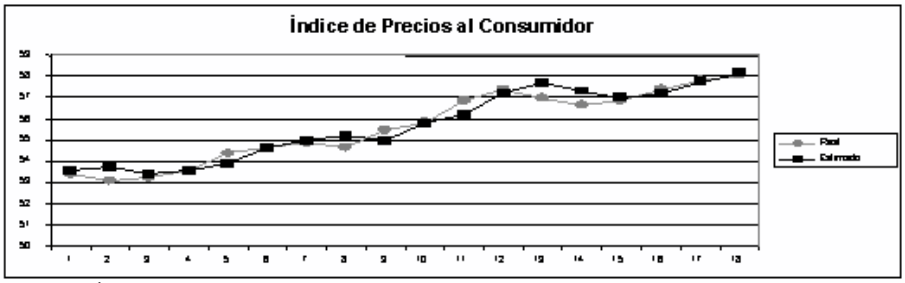

Fig. 4. Índice de Precios al Consumidor (IPC) y su aproximación utilizando *PGL*.

8. CONCLUSIONES Y TRABAJOS FUTUROS

En este trabajo se utiliza PGL para la obtención de modelos que permitan realizar estimaciones de series de tiempo, como los indicadores económicos. Los resultados obtenidos utilizando modelos hallados con PGL muestran la eficacia de los mismos, logrando mejores predicciones que los métodos tradicionales.

Estos alentadores resultados experimentales utilizando *Programación Genética Lineal* podrían de hecho esperarse, dada de la posibilidad de obtener modelos de predicción tanto lineales, como no lineales, utilizando PGL, lo que redunda en modelos más eficientes. Realizando una buena elección de parámetros para una corrida, se pueden encontrar tanto soluciones que representen a funciones lineales como a otros modelos que presenten una relación no lineal entre las variables de interés, lográndose en general mejores predicciones.

Como trabajos futuros se planea continuar los experimentos analizando otras series de interés, así como su comparación con otras técnicas como las Redes

Neurales. Otra área de investigación prometedora es la implementación de PGL con un enfoque multi-objetivo que considere diversos escenarios de entrenamiento y validación.

REFERENCIAS

1. Allen F., Karajalainen F.: Using genetic algorithms to find technical trading rules. J. Finan. Econom. 51, 245 -- 271 (1999).
2. Santini M., Tettamanzi A.: Genetic Programming for Financial Time Series Prediction, In P.L. Lanzi et al. (eds.), Proceeding of EuroGP 2001. LNCS vol. 2038, pp. 361 -- 370. Springer, Berlin (2001).
3. Bhattacharyya S., Pictet O., Zumbach G.: Knowledge-Intensive Genetic Discovery in Foreign Exchange Markets. In IEEE Transactions, vol. 6, no. 2, pp. 169 -- 181 (2002).
4. Brock W., Lakonishok J.,, LeBaron B.: Simple technical trading rules and the stochastic properties of stock returns. J. Finance. 47, 1731 -- 1764 (1992).
5. Zumbach G., Pictet O., Masutti O.:Genetic Programming with Syntactic Restrictions applied to Financial Volatility Forecasting. Olsen & Associates Research Institute for Applied Economics (2001).
6. Wooldridge J.: Introductory Econometrics: A Modern Approach. South-Western Thomson Learning, Sydney, Australia (1999).
7. Brameier M. ,Banzhaf W.: Linear Genetic Programming (Genetic and Evolutionary Computation), Springer, EEUU, Diciembre (2007).
8. Martinez F.: Análisis de las series temporales de los precios del mercado eléctrico mediante técnicas de clustering. Universidad de Sevilla – España. http://www.lsi.us.es/docs/doctorado/memorias/Martinez,%20Francisco.pdf
9. Koza J.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. Cambridge, MA: Mit Press (1992).
10. Nedjah N., Macedo L., Abraham A. (Eds.): Genetic Systems Programming: Theory and Experiences. Studies in Computational Intelligence Vol. 13 Springer, Berlin (2006).
11. Brameier M., Banzhaf W.: A comparision of linear genetic programming and neural networks in medical data mining. In IEEE Transactions, vol. 5, pp.17 -- 26 (2001).
12. Wagner N., Michalewicz Z., Khouja M., Mcgregor R.: Time Series Forecasting for Dynamic Environments: The DyFor Genetic Program Model Evolutionary Computation. IEEE Transactions on Evolutionary Computation, Vol. 11, No. 4., pp. 433 -- 452 (2007).
13. Hillier L.: Investigación de Operaciones. Séptima edición. McGraw Hill, México (2002).
14. Informe de Inflación, Gerencia de Estudios Económicos del Banco Central del Paraguay, http://www.bcp.gov.py/gee.