Evolutionary Computation to Estimate Volatility

Luis Alberto Navarro Huamani
luisnavarro@uni.edu.pe
Universidad Nacional de Ingenieria -Lima-Peru

Abstract– ***A performance evaluation study is implemented between the methods of Genetic Algorithms with Floating Point representation and some traditional optimization methods, in the task of estimating the parameters of a GARCH (1,1) Normal process, using artificial data obtained by simulation. The results show that the approximate solutions obtained by means of Genetic Algorithms present a better stability and precision with respect to the traditional optimization methods. The choice of the initial point in numerical optimization methods is not a critical condition in the use of Genetic Algorithms as a method to find the solution. Finally, Genetic Algorithm method is illustrated in the finding of the solution of the vector of parameters of the likelihood function of a GARCH (1,1) t-Student model, using data of rates of exchange returns of the Sol against to the Dollar.***

Keywords- Genetic Algorithms, Statistical Inference, GARCH

Digital Object Identifier (DOI):http://dx.doi.org/10.18687/LACCEI2018.1.1.436
ISBN: 978-0-9993443-1-6
ISSN: 2414-6390

Computación Evolucionaria para Estimar Volatilidad

Luis Alberto Navarro Huamaní
luisnavarro@uni.edu.pe

Universidad Nacional de Ingeniería -Lima-Perú

15 de Marzo, 2018

Abstract

Un estudio de evaluación de desempeño es implementado entre los métodos de Algoritmos Genéticos con representación Punto Flotante y algunos métodos de optimización tradicionales, en la tarea de estimar los parámetros de un proceso GARCH(1,1) Normal, usando datos artificiales obtenidos por simulación. Los resultados muestran que las soluciones aproximadas obtenidos mediante Algoritmos Genéticos presentan una mejor estabilidad y precisión con respecto a los métodos de optimización tradicionales. La elección del punto inicial en los métodos de optimización numéricos no es condición crítica en el uso del los Algoritmos Genéticos como método para hallar la solución. Finalmente, se ilustra el uso del método de Algoritmos Géneticos en el hallazgo de la solución del vector de parámetros de la función de verosimilitud de un modelo GARCH(1,1) *t*-Student, usando datos de tasas de retornos de intercambio del Sol frente al Dólar.

palabras claves Algoritmos Genéticos, Inferencia Estadística, GARCH

1 Introducción

Después de la introducción de los modelos Autoregresive Conditional Heteroskedasticity (ARCH) en los últimos 30 años diferentes extensiones y aplicaciones han sido propuestas. Diversas publicaciones académicas han aparecido en diferentes lugares y aplicados en muchas diferentes configuraciones, ver por ejemplo Bollerslev [1]. El interés en el uso de los modelos ARCH / GARCH se remite en explicar y pronosticar volatilidad. En particular, el fenómeno de volatilidad se manifiesta en los diversos instrumentos financieros negociados en diversos mercados financieros existentes en el mundo. Esto último conlleva a una demanda de buenos modelos para explicar y pronosticar volatilidad.

El interés en el uso de nuevos métodos para hallar soluciones aproximadas en un problema de optimización numérica, comprende también en estudiar su uso en la medición de volatilidades de un conjunto de instrumentos financieros. En particular, hallar la solución óptima de la función de verosimilitud de un modelo de volatilidad para un conjunto de instrumentos financieros, conlleva a elegir un método de aproximación con ciertas exigencias de precisión y estabilidad. En esa dirección, han aparecido estudios sobre el uso de algunas técnicas de inteligencia artificial para colocar algunos de éstos métodos como una alternativa para hallar soluciones aproximadas al problema de optimización generado por la función de verosimilitud de un modelo de volatilidad, ver por ejemplo Rizzo [8] que usó Algoritmos Genéticos, pero que no llevó a cabo una evaluación de desmpenõ; y Hung [3] quien utilizó Sistemas Difusos en la solución

El presente trabajo consiste en llevar a cabo una evaluación de desempeño entre los métodos de Algoritmos Genéticos con representación Punto Flotante y algunos métodos de optimización tradicionales, en la tarea de estimar los parámetros de un proceso GARCH(1,1) Normal. En la parte final, se implementa un algoritmo de computación evolucionaria para obtener una solución aproximada de la función de verosimilitud obtenida del modelo GARCH(1,1) *t*-Student, usando un conjunto de datos reales de la tasa de intercambio del Nuevo Sol versus el Dólar americano en un determinado periodo de tiempo.

Digital Object Identifier (DOI): http://dx.doi.org/10.18687/LACCEI2018.1.1.436
ISBN: 978-0-9993443-1-6
ISSN: 2414-6390

1.1 Modelo GARCH

Los modelos GARCH con componente de Regresión corresponden a una forma particular de heterocedasticidad. Un modelo heterocesdastico multiplicativo denominado de GARCH Normal puede ser escrito como:

$$\begin{aligned} y_t &= x_t^{'}\beta + u_t \;\; ; \;\; u_t|I_{t-1} \;\tilde{}\; N(0,\sigma_t^2) \quad (1) \\ \sigma_t^2 &= h(z_t,\gamma) \end{aligned}$$

este modelo es complicado por el hecho que $z_t = (y_{t-1} - x_{t-1}^{'}\beta)^2$, en donde como se puede observar no existe una forma de separar los parámetros regresivos de los parámetros de volatilidad.

Un modelo GARCH(1,1) Normal sin componente Regresivo es dado por

$$\begin{aligned} y_t &= \epsilon_t\sqrt{h_t} \;\; ; \;\; \epsilon_t \;\tilde{}\; N(0,1) \qquad (2) \\ h_t &= w + \alpha y_{t-1}^2 + \beta h_{t-1} \end{aligned}$$

en donde los ϵ_t son independientes y con distribución Normal de media cero y varianza unitaria. Se puede concluir que y_t, dada la información pasada I_{t-1}, tiene distribución Normal con media cero y varianza h_t. Definamos al vector de parámetros θ como $\theta = (w, \alpha, \beta)$. La varianza inicial h_0 es tratada como una constante conocida. Los parámetros de la ecuación de la varianza son retringidos por

$$w \geq 0, \;\; \alpha > 0, \;\; \beta \geq 0$$

para asegurar que h_t sea positivo. Otra restricción a ser considerada para que el proceso GARCH(1,1) Normal sea estacionario, es que

$$0 < \alpha + \beta < 1$$

Un modelo mas elaborado que el GARCH(1,1) Normal univariado que capture mejor las colas anchas de las distribuciones empíricas de los retornos de series financieras, es dado por el modelo GARCH(1,1) t-Student,

$$\begin{aligned} y_t &= \epsilon_t\sqrt{h_t} \;\; ; \;\; \epsilon_t \;\tilde{}\; t(0,1,\nu) \qquad (3) \\ h_t &= z_t^{'}\theta \end{aligned}$$

en donde $z_t^{'} = (1, y_{t-1}^2, h_{t-1})$ y el vector de parámetros θ se incrementa el nuevo parámetro componente ν; es decir, aquí $\theta = (w, \alpha, \beta, \nu)$. Una condición de estacionariedad mas débil (ver Nelson [7]), es dado por

$$\beta < 1$$

1.2 Estimación del modelo GARCH(1,1)

En esta parte consideraremos el proceso de estimación de los parametros involucrados en el modelo GARCH(1,1) a través del principio de máxima verosimilitud. Sea $\underline{y} = (y_1, ..., y_t, ..., y_T)$ el conjunto de T observaciones y θ el vector de parámetros.

En primer lugar consideremos el proceso GARCH(1,1) Normal con $\theta = (w, \alpha, \beta)$, según la ecuación (1). El modelo anterior descrito en términos de distribución es dado por

$$\begin{aligned} y_t &= \epsilon_t\sqrt{h_t} \;\; ; \;\; y_t|I_{t-1} \;\tilde{}\; N(0,h_t) \quad (4) \\ h_t &= z_t^{'}\theta \end{aligned}$$

en donde $z_t^{'} = (1, \epsilon_{t-1}^2, h_{t-1})$ y $\theta = (w, \alpha, \beta)$. Para una muestra de T observaciones $\underline{y} = (y_1, ..., y_T)$, la función de verosimilitud $l(\theta|\underline{y}) = p(\underline{y}|\theta)$ es

$$\begin{aligned} l(\theta|\underline{y}) &= \prod_{t=2}^{T} p(y_t|y_{t-1},\theta)p(y_1|y_0,\theta) \quad (5) \\ &\propto \prod_{t=1}^{T} (h_t)^{-\frac{1}{2}} e^{-\frac{1}{2}\frac{y_t^2}{h_t}} \quad (6) \end{aligned}$$

El logaritmo de la función de verosimilitud es proporcional a

$$L(\theta|\underline{y}) = \sum_{t=1}^{T} l_t(\theta) \qquad (7)$$

$$l_t(\theta) = -\frac{1}{2}log(h_t) - \frac{1}{2}y_t^2 h_t^{-1} \qquad (8)$$

Por otra parte, para el modelo GARCH(1,1) t-Student de la ecuación (3), el vector de parámetros es $\theta = (w, \alpha, \beta, \nu)$. El modelo anterior descrito en términos de distribución es dado por

$$\begin{aligned} y_t &= \epsilon_t\sqrt{h_t} \;\; ; \; y_t|I_{t-1} \;\tilde{}\; t(0, h_t\frac{\nu}{\nu+1}, \nu) (9) \\ h_t &= z_t^{'}\theta \end{aligned}$$

en donde $z_t^{'} = (1, y_{t-1}^2, h_{t-1})$ y $h_t\frac{\nu}{\nu+1}$ es el parámetro de escala. Para una muestra de T observaciones $\underline{y} = (y_1, ..., y_T)$, la función de verosimilitud es

$$\begin{aligned} l(\theta|\mathbf{y}) &= \prod_{t=2}^{T} p(y_t|y_{t-1},\theta)p(y_1|y_0,\theta) \qquad (10) \\ &\propto \prod_{t=1}^{T} \frac{\Gamma(\frac{\nu+1}{2})}{\Gamma(\frac{\nu}{2})}(\nu h_t)^{-\frac{1}{2}} \left[1 + \frac{y_t^2}{\nu h_t}\right]^{-\frac{\nu+1}{2}} \qquad (11) \end{aligned}$$

De manera similar, el logaritmo de la función de verosimilitud es

$$L(\theta|\underline{y}) \propto \sum_{t=1}^{T} l_t(\theta) \tag{12}$$

$$l_t(\theta) = log\frac{(\frac{\nu+1}{2})}{(\frac{\nu}{2})} - \frac{1}{2}log(\nu h_t) - \frac{\nu+1}{2}\left[1 + \frac{y_t^2}{\nu h_t}\right] \tag{13}$$

Según el principio de verosimilitud, tenemos que maximixar $L(\theta|\underline{y})$ con respecto de θ, donde $\theta = (w, \alpha, \beta)$ ó $\theta = (w, \alpha, \beta, \nu)$ según sea el caso. Por ejemplo, el formato de maximización para el caso de una distribución *t*-Student de $L(\theta|\underline{y}) = L(w, \alpha, \beta, \nu|\underline{y})$ es el siguiente:

$$\begin{array}{ll} \underset{(w,\alpha,\beta,\nu)}{\text{Max}} & L(\theta|\underline{y}) = L(w, \alpha, \beta, \nu|\underline{y}) \\ \text{S.A} & \beta < 1 \end{array}$$

Los valores de (w, α, β, ν) serán limitados mediante cotas. Se establece que $w \geq 0,\ \alpha > 0,\ \beta \geq 0$ y $\nu > 0$ para que la varianza h_t sea positiva.

Generalmente, la solución θ del problema de maximización es obtenida haciendo uso de métodos de optimización numérica restringidas. Un inconveniente en los métodos anteriores es que son algoritmos de busqueda local; es decir, sensible a la selección del punto inicial. Además, los métodos de optimización tradicionales involucran el conocimiento de las primeras y segundas derivadas de la función objetivo. Entre las teécnicas mas comunes que trabajan bajo este requerimiento tenemos a el BHHH y el gradiente-mixto descritos en Bollerslev ([1]) y Fiorentini ([2]) respectivamente. Ambos procedimientos son algoritmos alternativos entre muchas otros existentes. La implementación de estos algoritmos pueden ser encontrados en diversos Software Estadísticos como por ejemplo el Econometrics Views (Eviews) y el S-Plus.

2 Optimización Numérica

En la presente sección se evalúa el desempeño de un método de optimización tradicional y el método de Algoritmos Genéticos en un modelo GARCH(1,1) Normal, utilizando para ello un conjunto de datos generados por simulación estocástica por modelo. La intención en usar datos artificiales es comparar resultados de convergencia y estabilidad en las soluciones en las dos alternativas propuestas. Enseguida utilizaremos otro modelo GARCH(1,1) *t*-Student para estimar volatilidad en la serie de la tasa de intercambio del Nuevo Sol versus el Dólar americano en un determinado periodo de tiempo.

2.1 Algoritmos Genéticos

Los Algoritmos Genéticos representan una alternativa mucho mas barata para la comunidad estadística con respecto a las técnicas de optimización tradicionales. En particular, en las técnicas de optimización numéricas tradicionales se necesitan de las primeras derivadas y/o segundas derivadas que no siempre es de fácil derivación para obtener la solución aproximada a un problema, mientras que los Algoritmos Genéticos de representación punto flotante, descritos en Michalewicz [4], no necesitan de tales requerimientos en la implementación de dicha solución. Uno de los inconvenientes encontrados por las técnicas de optimización clásica es que son de búsqueda local y necesitan en general de un punto inicial significativo (cercano al óptimo) para que el algoritmo converja de manera adecuada. Las teécnicas de computación evolucionaria es de búsqueda de solución en paralelo, usando una población de soluciones posibles en el espacio de búsqueda hasta encontrar la solución óptima.

La representación binaria es la que tradicionalmente ha sido utilizada en la aplicación de los Algoritmos Genéticos. Sin embargo, el "Paralelismo implícito" que ofrece de codificación binaria no necesariamente dependen de usar este alfabeto binario, y es posible trabajar con alfabetos mayores con nuevos operadores genéticos. En particular, para el presente problema de optimización se manipula con variables en domínios contínuos, se trabaja con genes de códigos reales y con operadores genéticos especialmente desarrollados para ellos descritos en Michalewicz [4].

La representación de Punto Flotante, en lugar de Binaria, permite una cercanía entre el espacio original del problema y el espacio de representación, permitiendo una implementación mucho más fácil y eficiente de operadores dinámicos y cerrados.

2.1.1 Afinamiento Local

Los AG muestran dificultades inherentes en ejecutar busquedas locales para aplicaciones numéricas. La búsqueda local requiere la utilización de cromosomas binarios de mayor orden. Adicionalmente, existen problemas donde

los domínios de los parámetros son ilimitados, el número de parámetros es demasiado grande, y una precisión muy alta es requerida. Estos requerimientos implican que la longitud de un vector solución (binario) es bastante significante (para 100 variables con domínios en el rango [-500,500], y con precisión de seis dígitos después del punto decimal, la longitud del vector solución binario es 3000). Como habíamos mencionado anteriormente el desempenõ de los AG es pobre.

los Algoritmos Géneticos de representación Punto Flotante mejora la capacidad de afinamiento local para tratar problemas en donde se requiera alta precisión, al diseñar un operador de mutación especial cuyo desempeño es diferente del binario. El operador de mutación Binario cambia un bit de un cromosoma en cada iteración, y tal cambio propone solamente conocimiento local. Si el bit esta localizado al lado izquierdo de una secuencia que codifica a una variable, el impacto es significante en la magnitud absoluta; mientras que si el bit esta localizado al lado derecho de una secuencia, esta tiene poco impacto al aplicar el operador de mutación. Es decir, este conocimiento global posicional se utilizó para para desarrollar un nuevo operador de mutación que incorpore esta conocimiento específico del problema. El nuevo operador mutación usa la información posicional de la siguiente manera: cuando la población envejezca, los bits localizados a la derecha de cada secuencia que codifica a una variable tendrá mayor probabilidad de ser mutado, mientras que aquellos del lado izquierdo tendrán una probabilidad decreciente. En otra palabras,tal mutación ejecuta una búsqueda global del espacio de búsqueda al início del proceso iterativo pero una exploración local creciente a medida que aumentan las iteraciones del proceso de evolución. A este tipo de mutación se le denominará Mutación No-Uniforme.

2.1.2 La Representación

En la representación Punto Flotante cada cromosoma vector es codificado como un vector de números punto flotante de la misma longitud como el vector solución. Cada elemento es inicialmente seleccionado de tal manera que pertenezca al domínio deseado, y los operadores son cuidadosamente diseñados para preservar esta restricción.

La precisión de esta aproximación depende del tipo de maquina a utilizar para el procesamiento del programa de evolución, siendo que en general es mejor que la representación binaria. Esto porque es siempre posible extender la precisión de la representación binaria introduciendo mas bits, lo que ocasionaría lentitud en el algoritmo genético. la representación Punto Flotante es capaz de representar domínios muy grandes. Por otro lado, la representación binaria debe sacrificar precisión con un incremento en el tamaño del domínio, dado una longitud binaria fija. En la representación Punto Flotante es mucho mas fácil diseñar herramientas especiales para manejar restricciones no triviales.

2.1.3 Operadores Especializados

Antes de definir los nuevos operadores a ser utilizados en la representación punto flotante, asumiremos que tenemos el siguiente problema de optimización:

$$optimizar \quad f(x_1, ..., x_q) \in \Re^q$$

donde $< x_1, ..., x_q > \in D \subseteq \Re^q$ y D es convexo.

El domínio D es definido por los rangos de variables ($l_k \leq x_k \leq r_k$ para $k = 1, ..., q$) y por un conjunto de restricciones C. A partir de la convexidad del conjunto D se sigue que para punto $< x_1, ..., x_q >$ en D existe un rango feasible $< l_k, u_k >$ de una variable x_k ($k = 1, ..., q$), en donde otras variables x_i ($i = 1, ..., k-1, k+1, ..., q) \in D$ permanecen fijos.

Los operadores utilizados son bastantes diferentes de los usados para codificación binaria, dado que ellos trabajan en espacios diferentes. Sin embargo, debido a la similitud en los algoritmos, se define a los siguientes operadores:

Mutación Uniforme Definida de manera similar a la versión binaria: si $\mathbf{x}_i^t =< x_1, ..., x_q >$ es un cromosoma, entonces casa elemento x_k tiene igual chance de ser mutado. El resultado de una aplicación de este operador es un vector $\mathbf{x}_i^t =< x_1, ..., x_k^{'}, ...x_q >$, con $1 \leq k \leq q$, y $x_k^{'}$ un valor aleatorio del domínio del parámetro correspondiente.

Mutación No-Uniforme Es uno de los operadores responsables por la capacidad de afinamiento fino del sistema. Es definida como sigue: si $\mathbf{s}_x^t =< x_1, ..., x_q >$ es un cromosoma y el elemento x_k es seleccionado para esta mutación (el domínio de x_k es $[l_k, u_k]$), el resultado es un vector $\mathbf{s}_x^{t+1} =<$

$x_1, ..., x_k^{'}, ..., x_q >$, con $k \in 1, ..., q$, y

$$x_k^{'} = \begin{cases} x_k + \triangle(t, u_k - x_k) \ ; & \text{dígito aleatorio es 1} \\ x_k - \triangle(t, x_k - l_k) \ ; & \text{dígito aleatorio es 0} \end{cases} \quad (14)$$

donde la función $\triangle(t, y)$ retorna un valor en el rango $[0, y]$ tal que la probabilidad de $\triangle(t, y)$ esté cercano a 0 a medida que t se incremente. Esta propiedad hace que el operador busque el espacio uniformemente al inicio (para pocas iteraciones iniciales), y muy localmente en iteraciones mayores. Una función a usar para que esto se cumpla es :

$$\triangle(t, y) = y(1 - r^{(1-\frac{t}{T})^b})$$

donde r es un número aleatorio en [0,1], T es el número máximo de generaciones, y b es un parametro del sistema determinando el grado de no uniformidad.

Mutación Boundary Este operador requiere un único pariente $\mathbf{x}$ y produce un único descendiente $\mathbf{x}^{'}$. El operador es una variante de la mutación uniforme con $x_k^{'}$ tomando una de las cotas del domínio de x_k, $[l_k, u_k]$, con igual probabilidad.

Crossover Aritmético Este operador binario es definido como una combinación lineal de dos cromosomas: si $\mathbf{x}_1$ y $\mathbf{x}_2$ son cruzados, los descendientes resultantes son $\mathbf{x}_1^{'} = a.\mathbf{x}_1 + (1-a).\mathbf{x}_2$ y $\mathbf{x}_2^{'} = a.\mathbf{x}_2 + (1-a).\mathbf{x}_1$. Este operador usa un valor aleatorio $a \in [0, 1]$, y garantiza la cerradura de $\mathbf{x}_1^{'}, \mathbf{x}_2^{'} \in D$

Crossover Simple Este operador binario es definido como sigue: si $\mathbf{x}_1 =< x_1, ..., x_q >$ y $\mathbf{x}_2 =< y_1, ..., y_q >$ son cruzados después de la k-ésima posición, los descendientes resultantes son: $\mathbf{x}_1^{'} =< x_1, ..., x_k, y_{k+1}, ..., y_q >$ y $\mathbf{x}_2^{'} =< y_1, ..., y_k, x_{k+1}, ..., x_q >$. Es posible que este operador produzca descendientes fuera del domínio D. Para evitar esto, usaremos la propiedad de espacios convexos de que existe $a \in [0, 1]$ tal que los descendientes tengan la siguiente forma $\mathbf{x}_1^{'} =< x_1, ..., x_k, y_{k+1}.a + x_{k+1}.(1-a), ..., y_q.a + x_q.(1-a) >$ y $\mathbf{x}_2^{'} =< y_1, ..., y_k, x_{k+1}.a + y_{k+1}.(1-a), ..., x_q.a + y_q.(1-a) >$, feasibles.

Crossover Heurístico Este es el único operador crossover por las siguientes razones: (1) usa los valores de la función objetivo para determinar la dirección de la búsqueda, (2) produce unicamente un descendiente, y (3) posiblemente no produzca descendiente.

El operador genera un único descendiente $\mathbf{x}_3$ a partir de dos parientes $\mathbf{x}_1$ y $\mathbf{x}_2$ de acuerdo a la siguiente regla:

$$\mathbf{x}_3 = r.(\mathbf{x}_2 - \mathbf{x}_1) + \mathbf{x}_2$$

donde r es un número aleatorio entre 0 y 1, y el pariente $\mathbf{x}_2$ no es "peor" que $\mathbf{x}_1$. Es decir, $f(\mathbf{x}_2) \geq f(\mathbf{x}_1)$ para problemas de maximización, y $f(\mathbf{x}_2) \leq f(\mathbf{x}_1)$ para problemas de minimización.

Es posible que este operador genere un descendiente que no es feasible (no pertenece a D). Si esto ocurre genere otro valor aleatorio r y a seguir otros descendiente. Si después de un número finito de intentos ningún descendiente propuesto cumple las restricciones, el operador no produce un descendiente feasible.

2.2 Evaluación de desempeño de los Algoritmos tradicionales

En esta sección presentaremos algunos resultados obtenidos de la implementación de una técnica de optimización clásica en un modelo GARCH(1,1) Normal. En las diferentes propuestas de solución numérica vía optimización tradicional se alerta al usuario a tomar ciertas consideraciones en la selección de los valores iniciales en el proceso de estimación de la función de verosimilitud que se obtiene del modelo GARCH(1,1) Normal **univariado**. En algunos casos, se puede obtener buenas estimaciones iniciales para procedimientos de optimización con restricciones usando el método de mínimos cuadrados como una primera aproximación. En nuestro caso, los valores iniciales son determinados primero ajustando una ecuación mínimo cuadrática simple para el modelo GARCH(1,1) transformado, ver propuesta en Fiorentini *Et Al* [2].

Es importante destacar que la estrategia propuesta por Fiorentiniy es comúnmente usada por diversos usuarios y en diversos software estadísticos. Esta estrategia de selección de punto inicial a veces nos puede inducir a errores en la solución final del procedimiento iterativo. A continuación comparamos las soluciones encontradas por el software Eviews y el MatLab usando un procedimiento de mínimos cuadrados no lineales. Como muestra la tabla 1. Una serie artificial de tamaño 150 es generada por un

proceso GARCH(1,1) con parámetros verdaderos $w = 0.1$, $\alpha = 0.4$ y $\beta = 0.4$ para evaluar el desempeño de ambos algoritmos. A partir de los resultados se observa que ambas librerías de optimización producen resultados diferentes para un mismo punto inicial $w_0 = 0.1$, $\alpha_0 = 0.1$ y $\beta_0 = 0.9$. Ambas rutinas utilizan el procedimiento de optimización restringida de mínimos cuadrados no lineal basados en el método de Levenberg-Marquardt. Esto último nos conlleva a pensar que tenemos un problema de precisión en la solución obtenida por ambos softwares.

valor verdadero	estimaciones	
	Eviews	MatLab
$w = 0.1$	0.168	0.1564
$\alpha = 0.4$	0.238	0.3487
$\beta = 0.4$	0.049	0.4907

Table 1: Punto inicial $w_0 = 0.1$, $\alpha_0 = 0.1$ y $\beta_0 = 0.9$

La tabla 2 nos muestra algunas estimaciones obtenidas para $\theta = (w, \alpha, \beta)$ para puntos iniciales diferentes usando el software Eviews. Se puede ver que el vector $\theta = (w, \alpha, \beta)$ solución del algoritmo numérico de miínimos cuadrados no lineales es sensible a la selección del punto inicial.

punto inicial	EVIEWS
θ_0	estimaciones (# generaciones) $[log(\hat{\theta})]$
$(0.1, 0.1, 0.9)$	(0.168, 0.238, 0.049) (34) [-101.9104]
$(0.1, 0.1, 5)$	(-0.2619, -1.9025, 4.9497) (55) [-7912.479]
$(0.1, 0.7, 0.5)$	(0.1684, 0.2383, 0.0499) (32) [-101.9104]

Table 2: Estimaciones de $\theta = (0.1, 0.4, 0.4)$ - EVIEWS

De manera similar, la rutina *Constr* de MatLab también son sensibles a la selección del punto inicial. Debemos advertir que los valores de la función de verosimilitud $l^*(\hat{\theta})$ utilizados en la rutina *Constr* del MatLab, son proporcionales al $log(\hat{\theta})$ utilizados en la rutina de optimización numérica del software Eviews.

punto inicial	MATLAB
θ_0	estimaciones (# generaciones) $[log(\hat{\theta})]$
$(0.1, 0.1, 0.9)$	(0.1564, 0.3487, 0.4907) (23) [-46.5492]
$(0.1, 0.1, 5)$	(0.4564, 0.2047, 0.2766) (55) [-50.6745]
$(0.1, 0.7, 0.5)$	(0.1940, 0.5756, 0.3451) (64) [-47.4472]

Table 3: Estimaciones de $\theta = (0.1, 0.4, 0.4)$ - MATLAB

Del experimento de simulación, se puede advertir que el critério de selección del punto inicial del vector $\theta_0 = (w_0, \alpha_0, \beta_0)$ por mínimos cuadrados ordinarios utilizado por el EVIEW para la obtención del vector solución θ, no garantiza resultados mas acurísticos que las que proporciona el MatLab.

Est. Máx. Veros.	Alg.Genético
[Generación] Estimación	$[L(\hat{\theta})]$
[1](1.0958, 1.2500, 0.2557)	-87.8880
[2](0.7271, 0.0001, 0.1188)	-54.8558
[4](0.7271, 0.0001, 0.0001)	-54.6819
[7](0.7258, 0.0236, 0.0001)	-53.9444
[11](0.7250, 0.1306, 0.0001)	-53.0524
[28](0.5593, 0.1304, 0.0890)	-52.1822
[32](0.4061, 0.5082, 0.1261)	-51.2373
[33](0.4214, 0.3123, 0.1446)	-50.8293
[38](0.2467, 0.5056, 0.2719)	-48.0895
[44](0.2304, 0.3747, 0.4042)	-47.1474
[52](0.1935, 0.3747, 0.4158)	-46.7668
[61](0.1852, 0.3724, 0.4307)	-46.6844
[70](0.1728, 0.3510, 0.4662)	-46.5873
[80](0.1680, 0.3498, 0.4745)	-46.5684
[90](0.1664, 0.3496, 0.4756)	-46.5641
[100](0.1664, 0.3533, 0.4755)	-46.5632

Table 4: Estimaciones de $\theta = (w, \alpha, \beta, \nu)$ vía Algoritmo Genético - caso GARCH(1,1) Normal, datos simulados

2.3 Evaluación de desempeño del Algoritmo Genético

En un problema de optimización restringida, la forma geométrica del conjunto de soluciones en $\Re^q$ es la característica mas crucial del problema, con respecto al grado de dificultad que probablemente se encuentre a la hora de intentar resolver un problema determinado.

La definición del problema de optimización para el proceso GARCH(1,1) Normal fue enunciado en la sección anterior por la ecuación (7), y los detalles del algoritmo Genético con representación punto Flotante fuerón indicados en el trabajo monográfico de Navarro [6].

La tabla 4 nos muestra las estimaciones obtenidas para $\theta = (w, \alpha, \beta, \nu)$ para los datos de tasa de intercambio del Nuevo Sol versus el Dólar americano de un experimento en particular. Puede observarse que el Algoritmo Genético converge al valor de -46.5632 después de 100 iteraciones aproximadamente, y cuyo valor es mayor que las soluciones hallados por los métodos tradicionales de optimización

3 Algoritmos Genéticos para serie de retornos de n/s versus \$

En esta parte utilizaremos el modelo GARCH(1,1) t-Student para explicar la volatilidad de la serie de retornos diarias de la tasa de intercambio del Nuevo Sol versus el Dólar americano para los años de 1996 hasta 1998 (Fuente: Superintendencia de Banca y Seguros, Gerencia de Estudios Económicos, Departamento de Estadística y Estudios de Coyuntura), ver Navarro [5] Sea θ el vector de parámetros (w, α, β, ν), y para una muestra de T observaciones $\mathbf{y} = (y_1, ..., y_T)$ en donde y_t representa los retornos ajustados por su media muestral, la función de verosimilitud es dada por la ecuación (11) y el logaritmo de de la función de verosimilitud es dado por la ecuación (12). El formato de maximización de la función $L(\theta|\underline{y}) = L(w, \alpha, \beta, \nu|\underline{y})$ es el siguiente:

$$\begin{array}{ll} \underset{(w,\alpha,\beta,\nu)}{\text{Max}} & L(\theta|\underline{y}) = L(w, \alpha, \beta, \nu|\underline{y}) \\ \text{S.A} & \beta < 1 \end{array}$$

Los valores de (w, α, β, ν) serán limitados mediante cotas superiores e inferiores de manera apropiada para el programa de evolución a implementar. Se estableció que $w \geq 0,\ \alpha > 0,\ \beta \geq 0$ y $\nu > 0$ para que la varianza h_t sea positiva. También, se eligierón valores de $w_0,\ \alpha_0,\ \beta_0$ tales que $w_0 \geq w,\ \alpha_0 \geq \alpha,\ \beta_0 \geq \beta,\ \nu_0 \geq \nu$ para que nuestro espacio de búsqueda sea limitado.

La tabla 5 nos muestra algunas estimaciones obtenidas para $\theta = (w, \alpha, \beta, \nu)$ para los datos de tasa de intercambio del Nuevo Sol versus el Dólar americano de un experimento en particular. Se destaca que el Algoritmo Genético converge al valor de 629.5887 después de 100 iteraciones aproximadamente.

4 Conclusiones y Recomendaciones

El presente trabajo de investigación propone a los Algoritmos Genéticos como un método alternativo a los algoritmos de Optimización numéricos tradicionales, en la tarea de obtener una solución aproximada de un problema de estimación de la volatilidad modelado por un proceso GARCH(1,1) t-Student. Los resultados obtenidos en un experimento de simulación concluye que los Algoritmos Genéticos presentan mayor precisión y estabilidad en el hallazgo de la solución, con respecto a los métodos tradicionales utilizados en el presente estudio. Finalmente, un caso práctico con datos reales es implementado para la tasa de intercambio del Nuevo Sol versus el Dólar americano. La evaluación de desempeño para modelos más complejos debe ser una alternativa de trabajo de investigación para reforzar el uso de los Algoritmos Genéticos en el hallazgo de soluciones aproximadas, como por ejemplo trabajar con modelos ARCH/GARCH multivariados de mayor complejidad.

References

[1] Tim Bollerslev. Generalized Autoregressive Conditional Heteroskedasticity. *Journal of Econometrics*, 48:307–327, 1986.

[2] Calzolari Fiorentini and Panattoni. Analytic Derivates and the Computation of garch Estimates. *Journal of Applied Econometrics*, 48:399–417, 1996.

[3] Jui-Chung Hung. A fuzzy garch model applied to stock market scenario using a genetic

Est. Máx. Veros.	Alg. Genético
[Generación] Estimación	$[L(\hat{\theta})]$
[1](0.0306, 0.8556, 0.7504, 14.2493)	200.3257
[3](0.0306, 0.8556, 0.4004, 14.2493)	466.8456
[4](0.0306, 0.8556, 0.0001, 14.2493)	561.8244
[5](0.0029, 0.4596, 0.6059, 14.2493)	613.3035
[6](0.0029, 0.4596, 0.6049, 14.2493)	613.5115
[7](0.0029, 0.4596, 0.6049, 16.0225)	614.1492
[8](0.0029, 0.2761, 0.6049, 5.9341)	626.3757
[11](0.0029, 0.2419, 0.6049, 5.9341)	627.9509
[18](0.0029, 0.2417, 0.6049, 8.3027)	628.1353
[33](0.0029, 0.2197, 0.6389, 7.2029)	629.3044
[46](0.0029, 0.1992, 0.6561, 7.3202)	629.4646
[59](0.0028, 0.1998, 0.6710, 8.3717)	629.5795
[62](0.0028, 0.1991, 0.6710, 8.3717)	629.5810
[70](0.0028, 0.1991, 0.6698, 8.3717)	629.5836
[80](0.0028, 0.2013, 0.6680, 8.3717)	629.5872
[92](0.0028, 0.2019, 0.6676, 8.3717)	629.5879
[100](0.0028, 0.2019, 0.6676, 8.3724)	629.5887

Table 5: Estimaciones de $\theta = (w, \alpha, \beta, \nu)$ vía Algoritmo Genético - GARCH(1,1) t-Student, Datos reales

algorithm. *Expert Systems with Applications*, 36:11710–11717, 2009.

[4] Z Michalewicz. *Genetic Algorithms 'Data Structures = Evolution Programs*. Springer, 1996.

[5] Luis Navarro H. Estimación de la Volatilidad de la Tasa de Intercambio del Nuevo Sol versus el Dólar americano. Núcleo de Estadística Computacional (NEC) , *Reporte Técnico*, 2002.

[6] Luis Navarro H. Algoritmos Genéticos. Núcleo de Estadística Computacional (NEC), 2010.

[7] D.B. Nelson. Stationarity and Persistence in the GARCH(1,1) model. *Econometric Theory*, 6:318–334, 1990.

[8] Manuel Rizzo. On the choice of a genetic algorithm for estimating garch models. *Computational Economics*, 48:473–485, 2016.