Publicaciones en Ciencias y Tecnología
Vol 7, N^{0}1, Ene–Jun 2013, pp.23–35, ISSN:1856-8890, Depósito Legal: pp200702LA2730

MINIMIZACIÓN DE UNA FUNCIÓN DE ORDEN p MEDIANTE UN ALGORITMO GENÉTICO

[*]Rómulo Castillo Cárdenas [**] Adrian Rojas

Recibido: 01/10/2012 Aprobado: 10/06/2013

Resumen

En el presente trabajo consideramos el problema OVO (order value optimization), en el cual dadas m funciones continuas $f_1, ..., f_m$, definidas en un dominio $\Omega \in \mathbb{R}^n$ y un entero $p \in \{1, ..., m\}$, la funcion de orden p, con $p \leq m$, está dada por $f(x) = f_{i_p(x)}(x)$, para toda $x \in \Omega$, donde $i_p(x)$ es una función índice que satisface $f_{i_1(x)}(x) \leq f_{i_2(x)}(x) \leq ... \leq f_{i_p(x)}(x) \leq ... \leq f_{i_m(x)}(x)$. El problema que abordamos consiste entonces en minimizar f con $x \in \Omega$ por medio de un algoritmo genético que por su naturaleza intrínseca tiene la ventaja, sobre métodos de optimización continua existentes, de encontrar minimizadores globales. Ilustramos la aplicación de este algoritmo sobre ejemplos considerados mostrando su eficacia en la resolución de los mismos.

Palabras clave: Algoritmos genéticos, programación no lineal, problema OVO.

[*] *Departamento de Matemáticas. Decanato de Ciencias y Tecnología, Universidad Centroccidental Lisandro Alvarado, Barquisimeto, Venezuela, romulo@ucla.edu.ve, Parcialmente financiado por CDCHT-UCLA y proyecto (490045/2010-3) CNPq-PROSUL-Brasil.*

[**] *Departamento de Matemáticas. Universidad Nacional Experimental Rómulo Gallegos Calabozo, Venezuela, adrojasdiaz@hotmail.com, parcialmente financiado por misión Ciencias*

MINIMIZING A FUNCTION OF ORDER p USING A GENETIC ALGORITHM

Abstract

In this work we consider the problem OVO (order value optimization), where m given continuous functions $f_1, ..., f_m$, defined on a domain $\Omega \in \mathbb{R}^n$ and an integer $p \in \{1, ..., m\}$, the function of order p with $p \leq m$, is given by $f(x) = f_{i_p(x)}(x)$, for all $x \in \Omega$, where $i_p(x)$ is a function satisfying $f_{i_1(x)}(x) \leq f_{i_2(x)}(x) \leq ... \leq f_{i_p(x)}(x) \leq ... \leq f_{i_m(x)}(x)$. The problem we address is to minimize f with $x \in \Omega$ by a genetic algorithm that by its very nature has the advantage over existing continuous optimization methods, to find global minimizers. We illustrate the application of this algorithm on examples considered showing its effectiveness in solving them.

Keywords: Genetic algorithms, nonlinear programming, order value optimization.

Introducción

En optimización frecuentemente nos encontramos con el problema general de $\min_{x\in\Omega} f(x)$ donde $f : \Omega \subset \mathbb{R}^n \to \mathbb{R}$ es conocida como función objetivo y Ω es el conjunto factible. Un caso particular de este problema y de gran utilidad es el conocido problema min-max donde $f(x) = \min_{x\in\mathbb{R}^n} \max\{f_i(x) : i \in I\}$ y $f_i : \mathbb{R}^n \to \mathbb{R}$ para $i \in I$ son funciones dadas.
Este problema se generaliza mediante la introducción del problema OVO (Order Value Optimization). Dadas m funciones continuas $f_1, ..., f_m$, definidas en un dominio $\Omega \in \mathbb{R}^n$ y un entero $p \in \{1, ..., m\}$, la funcion de orden p, con $p \leq m$, se define mediante $f(x) = f_{i_p(x)}(x)$ para toda $x \in \Omega$, donde $i_p(x)$ es una funcion índice que satisface

$$f_{i_1(x)}(x) \leq f_{i_2(x)}(x) \leq ... \leq f_{i_p(x)}(x) \leq ... \leq f_{i_m(x)}(x).$$

Nuestro problema es entonces

$$\begin{array}{ll} \text{minimizar} & f(x) \\ & x \in \Omega\,, \end{array}$$

es decir, queremos minimizar la función que ocupa el lugar p-ésimo en cada x entre las m funciones dadas.

El problema involucra la minimización de una función que, en general es continua, pero no suave, aún en el caso en que las funciones involucradas sean diferenciables, mas aún, la introducción de hipótesis de convexidad en cada una de las funciones f_i para $i \in I$ no implica la convexidad de f.

Observe que para $p = m$ obtenemos el clásico problema min-max el cual si es convexo en el caso de que las funciones f_i sean convexas.

El problema OVO fue introducido y desarrollado recientemente en [1], [3], [2] y [4], en donde además de hacer una formulación más general como un problema de programación no-lineal con restricciones, introducen algunas formas de resolverlo usando métodos tipo Cauchy y del tipo Casi-Newton. Estos enfoques están basados en técnicas de optimización continua, necesitan el cálculo del gradiente de las funciones involucradas, y en en el segundo caso además, se utilizan aproximaciones de la matriz hessiana de la mismas, ver [6], sin embargo, lo que queremos enfatizar es que estas metodologías encuentran solamente soluciones locales, dependiendo del punto inicial usado en el problema considerado; este hecho constituye la razón principal del presente estudio. Motivación para este problema data, entre otras, de aplicaciones en problemas de toma de decisiones y problemas robustos de estimación de parámetros. Aplicaciones, en problemas de riesgo de valores pueden verse en [3], [4] y [7]. Una aplicación relevante sobre el alineamiento estructural de proteinas es mostrada en [5], donde es usado un método Gauss-Newton, ver [6], en la formulación del problema de superposición de dos estructuras mediante una suavización del problema OVO.

Por otra parte, conocemos que algoritmos genéticos y evolutivos han sido usados con eficiencia en diversidad de problemas, ver [8], [10], [11] y [9], principalmente cuando no se conocen métodos eficientes para su resolución. Son aplicables por tanto a problemas de optimización con restricciones y sin restricciones. Una breve descripcón de los mismos la haremos en la siguiente sección, sin embargo, en este trabajo particularizaremos en su aplicación al problema OVO antes mencionado, que por su naturaleza, puede ser adaptado y resuelto usando cualquier tipo de algoritmo genético; cabe destacar, que no es nuestro objetivo el comparar el desempeño numérico de diversas formulaciones de algoritmos genéticos ni evolutivos en la solución del problema planteado, sino que, considerando que los métodos usados de optimización continua para resolver el problema OVO, obtienen simplemente minimizadores locales, aprovechamos la metodología inherente a los algoritmos genéticos para mostrar que es posible encontrar soluciones globales a un problema que tienen relevancia en distintas áreas. Observemos además que no se trata de un problema de minimización vectorial, como lo es en forma particular el problema de programación multiobjetivo, ver [8], en donde hay conflicto entre las funciones consideradas, este hecho no está presente en el problema OVO, aunque rutinas de ordenación de datos y formas sofisticadas de operadores genéticos usados en las implementaciones usadas en

programación multiobjetivo podrían usarse en el problema OVO.

Finalmente indicamos que no conocemos otras experiencias del uso de algoritmos genéticos en la resolución del problema OVO y sus aplicaciones. En la sección 2 presentamos algunas generalidades concernientes al problema OVO, en la sección 3 hacemos una breve reseña sobre los algoritmos genéticos, posteriormente, en la sección 4, mostramos el comportamiento del algoritmo genético utilizado para ilustrar la obtención de resultados en comparación con lo mostrado en [2]. Finalmente se presentan las conclusiones.

Generalidades

A fin de tener una idea geométrica del problema, mostramos la función de orden p para algunos específicos valores de p.
Consideremos las siguientes funciones $f_1(x) = -x^2+x+\frac{7}{4}$, $f_2(x) = exp(-x)$, $f_3(x) = x^2-2x+1$ y $f_4(x) = exp(x-1)$. En las siguientes gráficas se muestran con línea gruesa las funciones de orden 1, de orden 2, de orden 3 y de orden 4 respectivamente. Observe la continuidad pero no diferenciabilidad de la función de orden p para $p = 1, 2, 3$ y 4.

En [2], para resolver el problema OVO se formula un algoritmo tipo Cauchy, es decir, de máximo descenso, que involucra condiciones generales sobre el problema, las funciones consideradas y sus respectivos gradientes, por ello presentamos a manera de observaciones, algunas suposiciones hechas sobre el problema, pero que no serán necesarias en nuestro abordaje. Suponga que $\Omega \subset \mathbb{R}^n$ es cerrado y convexo, y $f_1, ..., f_m$ tienen derivadas parciales continuas en un conjunto abierto que las contiene. El hecho que Ω sea cerrado y convexo es usado por [2] en la condición de optimalidad, en la definición de los subproblemas del algoritmo principal y en las demostraciones de los Teoremas 2.2 y 2.3 allí mostrados. Denotamos $g_j = \nabla f_j$ de ahora en adelante. Para todo $x, y \in \Omega$, $j = 1, ..., m$, asumamos que

$$\|g_j(x)\|_\infty \leq c$$

y

$$\|g_j(y) - g_j(x)\|_\infty \leq L\|y - x\|_\infty$$

En consecuencia, por ser cada f_j convexa, para todos $x, y \in \Omega$, $j = 1, ..., m$,

$$\|f_j(y) - f_j(x)\|_\infty \leq c\|y - x\|_\infty \tag{1}$$

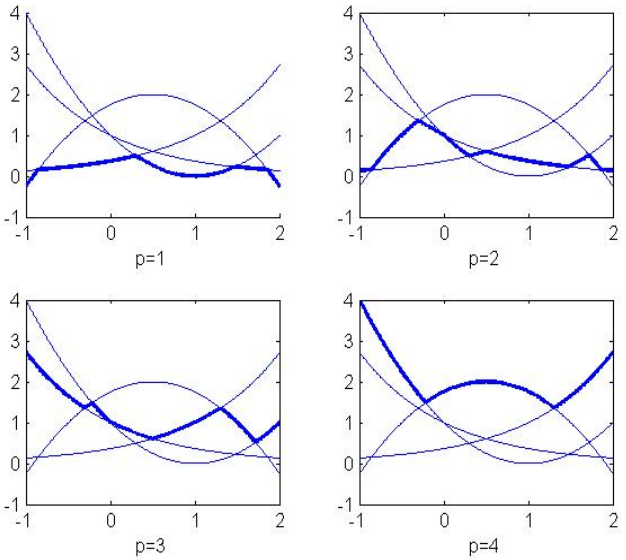

Figura 1: Función de orden p.

y

$$f_j(y) \leq f_j(x) + g_j(x)^T(y-x) + \frac{L}{2}\|y-x\|_\infty^2. \tag{2}$$

Dado $\varepsilon \geq 0$, $x \in \Omega$, definiremos

$$I_\varepsilon(x) = \{j \in \{1,...,m\} | f(x) - \varepsilon \leq f_j(x) \leq f(x) + \varepsilon\} \tag{3}$$

El teorema siguiente muestra la continuidad de la OVO-función, es tomado de [2] y lo mostramos por completitud.

Theorem 1 *La función p-orden-valor f es continua.*

Prueba. Sea la sucesión x^k tal que $x^k \to x$ y sin perder generalidad, para facilitar la notación identifiquémosla con una subsucesión de ella, supongamos por absurdo que para todo k se verifica

$$|f(x^k) - f(x)| \geq \delta > 0. \tag{4}$$

Para esta subsucesión, por definición de f, existe un índice $j \in \{1, ..., m\}$ de tal manera que

$$f(x^k) = f_j(x^k)$$

infinitamente muchas veces. Por lo tanto,

$$f_j(x^k) \geq f_\ell(x^k) \tag{5}$$

para al menos p índices $\ell \in \{1, ..., m\}$. Por otra parte,

$$f_j(x^k) \leq f_\ell(x^k) \tag{6}$$

para al menos $m-p+1$ índices ℓ en el conjunto $\{1, ..., m\}$. Como el número de subconjuntos de $\{1, ..., m\}$ es finito, un conjunto de índices ℓ que verifican (5) es repetido infinitamente en muchas ocasiones, y lo mismo sucede con un conjunto de índices ℓ que verifican (6). Por lo tanto, tomando límites en (5) y (6), obtenemos que

$$f_j(x) \geq f_\ell(x)$$

para al menos p índices $\ell \in \{1, ..., m\}$ y

$$f_j(x) \leq f_\ell(x)$$

para por lo menos $m - p + 1$ índices $\ell \in \{1, ..., m\}$.

Por lo tanto,

$$f(x) = f_j(x).$$

Pero $f_j(x^k) \to f_j(x)$, por lo que contradice (4).

El teorema siguiente, mostrado en [2] proporciona una condición necesaria para la optimalidad del problema OVO en base a la $\epsilon -$ *optimalidad*.

Definition 0.1 *Diremos que x es ε-óptimal si*

$$\mathrm{D} \equiv \{d \in \mathbb{R}^n \mid x + d \in \Omega \text{ y } g_j(x)^{\mathrm{T}} d < 0, \ \forall j \in I_\varepsilon(x)\} = \emptyset \tag{7}$$

Theorem 2 *Si $x_* \in \Omega$ es un minimizador local de $f(x)$ en $x \in \Omega$ y $\varepsilon \geq 0$, entonces x_* es ε-óptimal.*

Breve reseña sobre algoritmos genéticos

Los algoritmos genéticos constituyen técnicas computacionales basadas en heurísticas y fundamentos probabilísticos, deben su nombre a comparaciones biológicas de reproducción y adaptación de las especies vivientes al campo computacional. El problema dado es codificado y de allí es seleccionada una población inicial aleatoria de la cual se escogen padres, cada uno de los cuales representando una solución potencial del problema. A estos padres les son aplicados criterios de reproducción o cruzamiento, que involucran recombinaciones de los mismos y modificaciones aleatorias o mutaciones. Se obtienen así nuevos individuos o hijos que se evaluan mediante una función de adaptación o fitness relacionada con la función objetivo y se incorporan a una nueva población que será usada en el siguiente iterado del algoritmo genético. Diferentes algoritmos pueden obtenerse dependiendo de la técnica usada para representar los individuos (codificación) y los operadores genéticos de selección, recombinación y mutación.

En el presente trabajo no detallaremos la teoría sobre algoritmos genéticos ni evolutivos, estos últimos, constituyen una etapa de desarrollo posterior donde se hace énfasis en procedimientos estocásticos. Para una buena introducción y profundización de los mismos puede consultarse, por ejemplo [11],[9] y [10].

En nuestro caso, una vez formulado el problema OVO, observamos que es necesario un reordenamiento de los valores de las funciones involucradas en la función objetivo, para luego seleccionar cada elemento de la población que ocupe el lugar p-ésimo . Cada p-ésimo valor en el reordenamiento determina los elementos de la función objetivo OVO a ser minmizada, lo cual constituye una asignación directa una vez que se tiene la ordenación . El grado de dificultad dependerá del tipo de problema a resolver y las dimensiones del mismo, sin embargo, el uso de distintos operadores genéticos para obtener una solución satisfactoria no forma parte de lo que queremos mostrar, una comparación futura podría considerarse. Básicamente, la adaptación del problema OVO para ser resuelto con un algoritmo evolutivo dependerá de la representación computacional de la función objetivo, lo cual es inherente al problema, y la rutina de ordenamiento. En la siguiente sección, ilustramos con dos ejemplos la metodología usada, la función objetivo es un poco más elaborada para el segundo problema lo cual se aprecia en el código computacional.

Implementación Numérica

Consideramos dos implementaciones de un algoritmo genético en el cual usamos una codificacion o representación real de la población, además de usar operadores de reproducción, en el caso del cruzamiento: cruza $\beta X - \alpha$ y en el caso de la mutacion, mutacion Gaussiana. Otros distintos operadores pudieron ser usados, sin embargo, mostramos estos a manera de ilustración. El primer ejemplo es aplicado a minimizar la función de orden $p = 3$ con las funciones f_1, f_2, f_3 y f_4 mostradas en la figura 1, se muestra a continuación en una secuencia de 4 iteraciones no consecutivas que ilustran geométricamente el desarrollo del algoritmo en busca de la solución óptima la cual alcanza el valor de 1.7182 en 15 iteraciones.

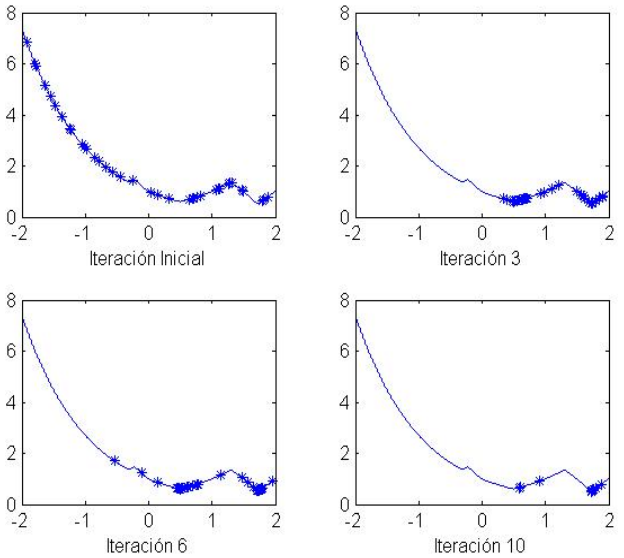

Figura 2: Ilustración gráfica.

El segundo ejemplo es un poco mas elaborado y muestra el buen desempeño del algoritmo genético en una aplicación de ajuste de curvas extraida de [2] donde se obtienen resultados similares.
Para la representación de los datos usamos codificación real (rutina pop2) a preferencia sobre la codificación binaria. La técnica usada para la esco-

gencia de la población es la selección proporcional con porcentajes para la reproducción y mutación indicados en el algoritmo general () los cuales son frecuentes en este tipo de estudios. Esto es hecho en la rutina select2 inmersa dentro de la rutina generareal2. La probabilidad de crossover y mutación efectuadas en la rutina cruzar y mutareal2, se aplican internamente en la rutina generareal2 a las columnas padre1 y padre2 de la población Pop para luego obtener dos vectores columnas con ambos hijos. La mutación gaussiana es efectuada en los nuevos individuos según rutina mutarealnormal. Anexamos a continuación el programa listado en matlab y hacemos algunos comentarios.

Programa Principal

```
% Algoritmo Genético para resolver el problema OVO
%==========================
clc;
clear;
lchrom = 20 % long. del cromosoma
popsize = 200*lchrom % tamaño de la población
pcross = 0.4 % prob.de crossover
pmutation = 0.0001 % prob. de mutación
maxgen = 86 % maxima cantidad de generaciones
p=20; % el valor del orden
while (p ≤ 43),
%Construir el intervalo MinInterv=-10; MaxInterv=10;
%Construir la población inicial
Pop =pop2(MinInterv,MaxInterv,popsize);
z = Pop;
Fitness=orden(z,p);
gen = 0;
while (gen ≤ maxgen),
gen = gen + 1;
[b,eli]=max(Fitness);
NewPop = generarealr2(Pop, Fitness, pcross, pmutation,MinInterv,
MaxInterv,p);  NewPop(:,1)=Pop(:,eli);
w = NewPop;
Fitness =orden(w,p);
[FitMax,maxf] = max(Fitness);
Pop = NewPop;
xval=NewPop(:,maxf);
```

```
end
xval.
```

La función orden p (p=1,2,...,46) para el siguiente ejemplo en un problema de ajuste de curvas es codificada como: % Función de orden p

```
function y = orden(x,p);
H=[f1(1,x);f1(2,x);f1(3,x);f1(4,x);f1(5,x);f1(6,x);f1(7,x);f1(8,x);f1(9,x);...
f1(10,x);f1(11,x);f1(12,x);f1(13,x);f1(14,x);f1(15,x);f1(16,x);f1(17,x);...
f1(18,x);f1(19,x);f1(20,x);f1(21,x);f1(22,x);f1(23,x);f1(24,x);f1(25,x);...
f1(26,x);f1(27,x);f1(28,x);f1(29,x);f1(30,x);f1(31,x);f1(32,x);f1(33,x);...
f1(34,x);f1(35,x);f1(36,x);f1(37,x);f1(38,x);f1(39,x);f1(40,x);f1(41,x);...
f1(42,x);f1(43,x);f1(44,x);f1(45,x);f1(46,x)];
% H es una matriz de valores de las m=46 funciones
J=sort(H,1,'ascend'); % J es una matriz de orden por columna de forma creciente bajo la instrucción sort de la matriz H
y=J(p,:); % y es una matriz fila con los valores de la fila p de la matriz J
y=-y;
```

Consideremos el modelo,

$$y(t,x) = x_1 + x_2 t + x_3 t^2 + x_4 t^3$$

para un conjunto de datos (t_i, y_i), $i = 1, ..., m$. Esta función corresponde a un modelo cúbico usado para ajuste de curvas entre otras aplicaciones. Consideremos a su vez las funciones error dadas por

$$f_i(x) = (y(t_i, x) - y_i)^2.$$

Mostramos resultados y conclusiones al minimizar la función de orden p para distintos valores del mismo a fin de dar una interesante interpretación.

Teniendo en cuenta la solución,

$$x^* = (x_1^*, x_2^*, x_3^*, x_4^*) = (0, 2, -3, 1),$$

generamos datos aleatorios usando,

$$w_i = y(t_i, x^*) \quad \text{con,}$$

$$t_i = -1 + 0{,}5i \quad \text{para} \quad i = 1, .., m, \text{ y } m = 46,$$

$$y_i = \begin{cases} 10 & \text{si} \quad 7 \leq i \leq 16 \\ w_i + r_i & \text{en otro caso,} \end{cases}$$

donde r_i es aleatorio entre -0.01 y 0.01. De esta forma, $y_7, ..., y_{16}$ simulan observaciones erradas o "outliers".

En el Cuadro 1 mostramos resultados de las corridas del algoritmo evolutivo considerado para minimizar la función de orden p para diversos valores del mismo, se observan los valores obtenidos para cada valor de p, de las incógnitas x_1^*, x_2^*, x_3^* y x_4^* y el correspondiente valor objetivo $fobj$. Remarcamos que la solución óptima es $x^* = (x_1^*, x_2^*, x_3^*, x_4^*) = (0{,}0004, 1{,}9998, -3{,}0000, 1{,}0000)$.

p	x_1	x_2	x_3	x_4	$fobj$
20	−0,1960	2,0016	−3,0027	1,0006	0,0000
21	0.2898	2.1330	-3.4764	1.1365	0.0137
22	0.2665	2.1107	-3.4446	1.1358	0.0187
23	0.2608	2.1527	-3.2985	1.0590	0.0351
24	0.2565	2.1771	-3.3712	1.0814	0.0262
25	0.2095	2.1742	-3.3330	1.0731	0.0289
26	0.1673	2.1149	-3.3061	1.0901	0.0326
27	0.1952	2.1474	-3.3540	1.1020	0.0351
28	-0.0031	2.0036	-3.0010	1.0000	0.0401
29	0.1958	1.6704	-2.8308	0.9729	0.0426
30	-0.0571	2.1659	-3.1272	1.0282	0.0439
31	0.0993	1.8389	-2.9201	0.9876	0.0413
32	-0.0224	2.0508	-3.0310	1.0054	0.0415
33	-0.0200	2.0361	-3.0176	1.0025	0.0413
34	0.0206	2.1133	-3.1126	1.0226	0.0629
35	0.0096	2.0153	-3.0202	1.0045	0.0446
36	0.0004	1.9998	-3.0000	1.0000	0.0402
37	8.5968	-8.1420	-0.9937	1.1061	9.3787
38	9.1908	-8.5327	-1.0137	1.1142	11.7662
39	9.9883	-9.3235	-0.8255	1.0960	14.4629
40	7.4122	-0.0932	-7.1706	2.2765	17.1600
41	6.1998	2.5339	-8.6074	2.5049	17.1637
42	6.2138	2.7182	-8.6283	2.4799	18.2686
43	5.8688	2.4626	-7.6065	2.1383	20.5587

Cuadro 1: Solución del problema OVO mediante el algoritmo AGRealOrdenp

Puede observarse que en general, los valores objetivos encontrados en comparación con el valor real tienen un error de 0.05 similares a los encontrados en [2] para el mismo problema, sin embargo, para los valores de $p = 27$ y $p = 34$ se obtuvieron resultados satisfactorios correspondientes a minimizadores globales, a diferencia de los resultados presentados en [2] en

el cual se mostraron resultados de minimizadores locales.

Conclusiones

Se ha presentado una versión de algoritmo genético para minimizar una función de orden p en la cual mostramos dos ejemplos ilustrativos de la eficacia del algoritmo. Recalcamos que versiones similares o más elaboradas de estos algoritmos incluyendo algoritmos evolutivos pueden ser utilizadas en la medida que la complejidad del problema a resolver sea mayor. No se hizo hincapié en mejorar el número de iteraciones del mismo ni el tiempo computacional pues el propósito fue el de establecer una metodología alterna para resolver el problema OVO, notando que, una ventaja de esta metodología es, a diferencia de los algoritmos de optimización continua usados en la literatura, la de encontrar minimizadores globales, lo cual en algunas aplicaciones es muy importante. En los resultados del ejemplo 2, a diferencia de los mostrados en [3] para el mismo problema, se observa que para los valores $p = 27$ y $p = 34$ encontramos minimizadores globales, lo cual se verifica puesto que en este ejemplo se conocía previamente el minimizador global. Cuando no se conocen previamente estos minimizadores, que es el caso común y general, la necesidad de conseguir una mejor aproximación de minimizadores globales es un problema común a los algoritmos genéticos y evolutivos en general para lo cual se han considerado diversas estrategias las cuales escapan al objetivo del presente trabajo pero que pueden consultarse, por ejemplo, en [9] y [10]. Futuras comparaciones de diferentes versiones de algoritmos genéticos que involucren el uso de operadores genéticos más eficientes podrían ser consideradas en problemas con mayor complejidad, como es el caso estudiado en [7], en donde la función OVO aparece en las restricciones.

Referencias

[1] Andreani R., Dunder C. y Martínez J. M., Nonlinear programming reformulation of the Order-Value Optimization problem, Tecnical Report, Institute of Mathematics, University of Campinas, Brasil, (2005).

[2] Andreani R., Dunder C. y Martíinez J. M., Order-Value Optimization: formulation and solution by means of a primal Cauchy Method, Mathematical Methods of Operation research **58**, , (2003), pp. 387–399.

[3] Andreani R., Martínez J. M., Martínez L. y Yano F., Low the Order-Value Optimization and aplications, Tecnical Report, Institute of Mathematics, University of Campinas, Brasil, (2007).

[4] Andreani R., Martínez J. M., Salvatierra M. y Yano F., Quasi-Newton Methods for Order-Value Optimization and Value-at-Ristk, Pacific Journal of Optimization **2**, pp. 11-33, (2006).

[5] Andreani R., Martínez J. M., Martínez L. y Yano F. S, Low Order-Value Optimization and new applications. Journal of Global Optimization 43, pp. 1-10 (2009).

[6] Bertsekas D., Nonlinear Programming: 2da Edition, Athena Scientific, (1999).

[7] Birgin E. G., Bueno L. F., Krejić N. y Martínez J. M., Low Order-Value Approach for Solving VaR-Constrained Optimization Problems, Journal of Global Optimization 51, pp. 715-742, (2011).

[8] Christian D. von Lucken M., Msc.Thesis. Algoritmos Evolutivos para Optimización Multiobjetivo. Universidad de la Asunción. Paraguay. (2003).

[9] Coello C. , Introducción a la computación Evolutiva. IEEE Computational Intelligence , vol. 1, No. 1, pp 28-36, (2006).

[10] Deb K. Multi-objetive Optimization Using Evolutionary Algorithms. John Wiley & Sons, Inc., New York, NY.(2001).

[11] Goldberg, D.E. , *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co. Inc., Boston, MA, USA. (1989).