Año 4, Vol. II, N° 7
ISSN: 1856-8327

Optimización Multiobjetivo Evolutiva a través un modelo de programación no – lineal por metas

Evolutionary Multi-objective Optimization by a model of non linear programming by goals

Jaquelin Loyo de Sardi

Palabras Clave: Algoritmos evolutivos, algoritmos genéticos, evolución natural, optimización multiobjetivo, programación por metas

Key Words: Evolutionary algorithms, genetic algorithms, natural history, multi-objective optimization, goal programming

RESUMEN

Los Algoritmos genéticos son técnicas heurísticas basadas en métodos denominados evolutivos por emular la evolución natural. En el trabajo aquí presentado se diseña un algoritmo evolutivo para resolver un problema de optimización multiobjetivo expresado como un modelo de Programación No Lineal por Metas con el objetivo de mostrar su desempeño. Se experimentó con dos ejemplos: toma de decisiones de inversión en un banco y, un problema numérico no lineal. Los resultados obtenidos han sido satisfactorios y muestran la habilidad de los Algoritmos Genéticos como herramienta de optimización en esta clase de problemas de Investigación de Operaciones.

ABSTRACT

Genetic algorithms are heuristic techniques based on evolutionary methods called evolutives to emulate natural evolution. In the presented work a genetic algorithm is designed to solve multiobjective optimization problems expressed as a Nonlinear Programming Goal Model with the purpose of showing their performance. It was experimented with two examples: making investment decisions on a bench and a nonlinear numerical problem. The results are satisfactory and show the ability of genetic algorithms as optimization tool for this class of problems of Operations Research.

Año 4, Vol. II, N° 7
ISSN: 1856-8327

INTRODUCCIÓN

La optimización con objetivos múltiples extiende la teoría de la optimización permitiendo que múltiples objetivos sean optimados simultáneamente. Ha sido usada durante años en economía (Takayama, 1974) y en ciencias gerenciales (Evans, 1984) y gradualmente se ha introducido en la ingeniería (Wu, 1995). Es conocida por diversos nombres tales como Optimización Pareto, Optimización Vectorial, Optimización Eficiente, Optimización Multicriterio y otros más. Las soluciones suministradas son denominadas Óptima Pareto, Vector Máximo, Puntos Eficientes y Soluciones No Dominadas.

Casi todos los problemas reales involucran la optimización de varios objetivos que a menudo están en conflictos y compiten entre sí. Mientras que en la optimización con un simple objetivo usualmente la solución óptima está claramente bien definida, no sucede lo mismo en problemas con objetivos múltiples. En lugar de un óptimo simple, existen un conjunto de alternativas conocidas como soluciones óptima Pareto. Éstas soluciones son óptimas en al amplio sentido de que ningunas otras soluciones son superiores a éstas cuando todos los objetivos son considerados.

Los problemas de optimización multiobjetivo son muy comunes en el área de la ingeniería y las técnicas para resolverlos muy diferentes a las empleadas en la optimización simple.

METODOLOGÍA

Problema de optimización multiobjetivo (OMO)

Un problema general de OMO incluye un conjunto de n variables de decisión, un conjunto de l funciones objetivo y un conjunto de m restricciones. Las funciones objetivos y las restricciones son funciones de las variables. La meta de la optimización es:

$$\text{maximizar} \quad \mathbf{y} = \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}),..., f_l(\mathbf{x})) \quad (1)$$

$$\text{sujeto a} \quad \mathbf{r}(\mathbf{x}) = (r_1(\mathbf{x}),r_2(\mathbf{x}),...,\mathbf{r}_m(\mathbf{x})) \leq 0 \quad (2)$$

$$\text{donde} \quad \mathbf{x} = (x_1,x_2,...,x_n) \in X \quad (3)$$

$$\mathbf{y} = (y_1,y_2,...,y_n) \in Y \quad (4)$$

y x es el vector de decisión, y es el vector objetivo. X denota el espacio de decisiones, y Y es denominado el espacio objetivo. Las restricciones $r(x) \leq 0$ determinan el conjunto de soluciones factibles.

Una solución factible x_1 para un modelo con objetivos múltiples se dice ser una solución no dominada o un punto eficiente, si no existe otra solución factible x_2 que satisfaga (Rardin, 1998):

1. x_2 tiene los mismos o mejores valores que x_1 para cada función objetivo en el modelo.
2. x_2 tiene estrictamente mejor valor que x_1 para al menos una función objetivo.

El conjunto de todas las soluciones óptimas eficientes es conocido como frontera eficiente o frontera Pareto óptima como se observa en la Fig. 1.

Año 4, Vol. II, N° 7
ISSN: 1856-8327

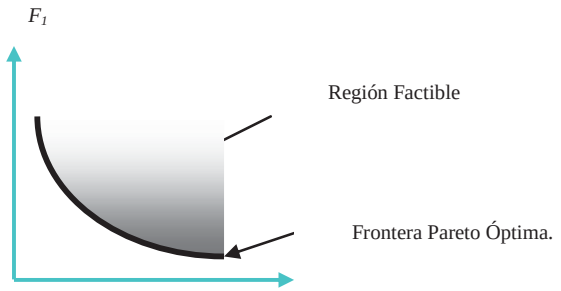

Figura 1. Frontera eficiente de optimización

Métodos tradicionales de la OMO

En un problema de OMO el objetivo es hallar el conjunto Pareto Óptimo o Frontera Eficiente. Para alcanzarlo, los métodos tradicionales agregan los objetivos en una simple y parametrizada función objetivo. Tales técnicas, por lo general, llevan a cabo varias corridas con diferentes parámetros de tal manera de obtener un conjunto de soluciones que se aproximen al conjunto Pareto Óptimo. Éstas técnicas son básicamente independientes del algoritmo de optimización subyacente.

Ejemplos de ésta clase de técnicas son: método de suma ponderada de objetivos, método de prioridades o restricciones, técnica minmax y programación por metas.

Programación no lineal por metas

La programación por metas es una técnica poderosa para resolver problemas de optimización multiobjetivo ideada por Charnes y Cooper (1961) que ha jugado además un rol clave en aplicaciones a problemas industriales. En la técnica se deben asignar metas que se desean alcanzar para cada objetivo. Estos valores son incorporados en el problema como restricciones adicionales. La función objetivo entonces tratará de minimizar las desviaciones de cada objetivo con respecto a su meta. La forma más simple de éste método puede ser formulada como:

$$\min \sum_{i=1}^{m} \left| f_i(\bar{x}) \quad _i \right|_i, \text{ sujeto a } \bar{x} \in f \qquad (5)$$

donde g_i denota el valor de la meta de la función objetivo y f representa la región factible.

Hay dos clases de problemas de programación por metas. No restrictiva en donde todas las metas son de importancia comparable. Restrictiva en donde existe una jerarquía de niveles de prioridades para las metas, de tal manera que las metas de importancia primaria reciben atención de primera prioridad, las de segunda importancia reciben atención de segunda prioridad y así sucesivamente. Por defecto el término programación por metas se refiere a éste último concepto.

Los problemas de programación no lineal por metas son muy importantes en la ingeniería ya que involucran múltiples objetivos y restricciones no lineales. Las técnicas para resolverlos fueron tratadas por Ignizio (1976), Hwanh y Masud (1979), y por Weistroffer (1983). Cuatro principales procedimientos son expuestos y discutidos por Saber y Ravindran (1993):

- Basados en el método Simplex.
- Búsqueda Directa.
- Búsqueda por Gradiente.
- Métodos Interactivos.

Año 4, Vol. II, N° 7
ISSN: 1856-8327

En vista de que los problemas de programación no lineal por metas pueden tener diferentes estructuras y grado de no linealidad, el desempeño de tales técnicas, en cuanto a confiabilidad, precisión, convergencia, preparación y esfuerzo de cómputo, dependerá del problema.

Formulación del problema

La formulación general de un modelo de programación no lineal por metas es de la siguiente manera:

$$\min z = \sum_{s=1}^{m}\sum_{r=1}^{n} P_s(\alpha_{sr}^{+}u_r^{+} + \beta_{sr}u_r) \quad (6)$$

$$s.t.\ f_r(x) \quad b_r = u_r^{+} \quad u_r \quad , r = 1,2,\ldots,n \quad (7)$$

$$g_r(x) \le 0, \quad r = n+1,\ldots,m_1(=n+m_1) \quad (8)$$

$$h_r(x) = 0, \quad r = m_1+1,\ldots,m(=m+m_1) \quad (9)$$

$$\alpha_r^{+}, \beta_r \ge 0, \quad r = 1,2,\ldots,m$$

Donde:

Ps es la prioridad restrictiva s (Ps>>Pk+1para toda s).
u_r+ es la desviación positiva sobre la meta r.
u_r- es la desviación negativa debajo de la meta r.
α^{+}_{sr} es la ponderación positiva asignada a u+r en la prioridad Ps.
β-$_{rr}$ es la ponderación negativa asignada a u-$_r$ en la prioridad Ps
f_r son las funciones objetivos.
g_r metas a alcanzar por cada objetivo.

Los métodos existentes para resolver ésta clase de problemas suelen dar resultados aceptables; sin embargo, existen limitaciones en su aplicación:
-Requieren la aproximación del modelo no lineal a un modelo lineal.
-La aproximación lineal es solamente una buena aproximación al objetivo lineal en las cercanías de una solución factible.
-Las aproximaciones lineales obtenidas suelen resultar en modelos que amplían el número de variables y de restricciones.
-A medida que crece el número de variables el proceso se complica, además se requiere que todas las funciones objetivos involucradas sean derivables.
Tales limitaciones han motivado la experimentación con técnicas de la computación emergente como los modernos algoritmos evolutivos.

Algoritmo genético (AG)

Los Algoritmos Genéticos son Modelos Computacionales de la Evolución Humana y son útiles tanto como métodos de búsqueda para la resolución de problemas como para modelar sistemas evolutivos (Forrest, 1996). Pertenecen a la clase de métodos probabilísticos independientes del problema que pueden manejar cualquier clase de funciones objetivo y de restricciones.
Debido a su naturaleza evolutiva los algoritmos genéticos realizan búsquedas robustas y multidireccionales en espacios complejos mediante el mantenimiento de una población de soluciones potenciales. De tal manera que ofrecen habilidad para manejar complejos problemas reales de optimización multiobjetivo formulados como modelos de programación no lineal por metas. La figura 2 muestra el proceso de un AG simple.

$f_j(x)$ =

```
Procedure Algoritmo Genético
Begin
        t <= 0
        inicializar p(t)
        evaluar p(t)
  while ( no condición terminación )do
  begin
       t<= t+1
     seleccionar p(t) de p(t-1)
      alterar p(t): recombina.
        mutar
        evaluar p(t)
  end

end
```

Figura 2. Algoritmo Genético Simple

Representación e Inicialización

Un individuo es un cromosoma el cual a su vez representa una solución, de tal forma que para el algoritmo a diseñar un cromosoma será un vector, cuyo tamaño dependerá de la cantidad de variables de decisión n, de la forma siguiente:

$$X = (x_1, x_2, x_3, \ldots\ldots\ldots x_n)$$

En donde cada x_i representa un gene del cromosoma o variable del problema específico. La información de cada individuo será manejada a través de una estructura que almacene además, otros datos de importancia asociados con la solución o individuo particular: aptitud o valor de la función objetivo, cota inferior y superior de cada gene, aptitud relativa y aptitud acumulada. El diseño de tal estructura concierne con las características del lenguaje de programación seleccionado.

La primera población puede ser generada aleatoriamente dentro de sus cotas o suministrada interactivamente por el usuario. Si el usuario decide introducir una o varias soluciones factibles que cumplan todas las restricciones, el proceso será más eficiente en el sentido de que cuando se apliquen los operadores de mutación y de cruce específicos, se garantizará generar una nueva población igualmente factible. De tal manera que el proceso de inicialización incluye la generación de una población de individuos soluciones dentro de sus cotas y la inicialización a cero de los valores de aptitud de cada miembro de la población inicial.

Función de Aptitud

La función de evaluación de la aptitud de cada individuo quedará establecida de la forma que se muestra:

$$\min z = \sum_{s=1}^{m}\sum_{r=1}^{n} P_s(\alpha_{sr}^{+}u_r^{+} + \beta_{sr}u_r) + f_{penalidades} \quad (10)$$

Esta función de evaluación puede ser representada también de la siguiente manera:

Donde $P_i >> P_{i+1}$ y, $X = (x_1, x_2, x_3, \ldots\ldots\ldots x_n)$

es un cromosoma o solución. La evaluación se lleva a cabo sobre cada individuo de la población. Un individuo o cromosoma con el menor valor de la función de evaluación tendrá la mejor aptitud dentro de la población considerada; por consiguiente la mejor solución será:

$$F(X) = P_1 \sum_{r=1}^{n} (\alpha_{1r}u_r^{+} + \beta_{1r}u_r^{-}) + P_2 \sum_{r=1}^{n} (\alpha_{2r}u_r^{+} + \beta_{2r}u_r^{-}) + \ldots$$

$$+ P_m \sum_{r=1}^{n} (\alpha_{mr}u_r^{+} + \beta_{mr}u_r^{-}) + \text{fpenalidad} \qquad (11)$$

Mínimo (min $F(x)_i$, i = 1.....número de generaciones)

Fpenalidad: representa una función que evalúa las penalizaciones que se le dan a las soluciones o individuos no factibles por no cumplir con alguna de las restricciones. Se tiene entonces que si no se viola ninguna de las restricciones esta función tendrá valor cero, de lo contrario se tiene que:

$$f_j(X) = \begin{Bmatrix} \max\{0, gj(x)\}, \text{si} & 1 \le j \le q \\ |hj(x)|, \quad \text{si} & q+1 \le j \le m \end{Bmatrix} \quad (12)$$

Siendo $g_j(x)$ las restricciones o inecuaciones ≤ y h_j las restricciones igualdad o ecuaciones. En el programa aquí desarrollado se emplea como función de penalidad la siguiente expresión:

$$Fpenalidad = C * t^{\alpha} \sum_{j=1}^{m} f_j^{\beta}(x) \qquad (13)$$

Donde, C, α, β son constantes con valor de 0.5, por ser éste una selección adecuada según reportes (Joines y Houck, 1994); t es un valor que representa el número de la generación correspondiente. De tal forma que la presión sobre individuo no factible es incrementada hacia el final de la simulación cuando t toma valores grandes, ya que el término $(C \times t)^{\alpha}$ se incrementa apreciablemente en las últimas generaciones.

Selección

El proceso de selección se llevará a cabo en la forma clásica "girar la rueda de la ruleta", procedimiento de selección basado en la distribución de probabilidades de los valores de aptitud; se habrá de incorporar además al método el modelo "elitista" de tal forma de asegurar que el mejor individuo factible sobreviva a través de las siguientes generaciones. Se asigna el mejor miembro de la generación previa como último elemento del arreglo población, si el mejor miembro de la población actual es peor que el mejor miembro de la generación previa, éste último reemplaza al peor miembro de la población actual.

Cruce

Se aplicará el cruce aritmético (Michalewicz, 1996), el cual es un operador binario definido como una combinación lineal de dos vectores y que garantiza obtener soluciones en el dominio del problema. El procedimiento general es el siguiente:

-Se establece una probabilidad de cruce **pc,** la que ayudará a determinar la cantidad promedio de individuos que se habrán de cruzar: pc x tamaño población.

-Para cada miembro de la población (cromosoma) se genera un número aleatorio a en el rango de [0..1], si a< pc se selecciona el cromosoma dado para cruce.

-Para cada par de cromosomas seleccionados (padres) se aplica el operador aritmético de cruce. Si c1 y c2 son los cromosomas a cruzar, se generarán

dos nuevos miembros (hijos) de la forma que se muestra:

$$c_1' = b.\ c_1 + (1\text{-}b)\ .\ c_2 \qquad (14)$$

$$c_2' = b.\ c_2 + (1\text{-}b)\ .\ c_1 \qquad (15)$$

donde b€[0..1], por lo que se asegura generar hijos en el dominio del problema.

Mutación

Se aplicará el operador de mutación no uniforme (Michalewicz, 1996) de acuerdo con el siguiente procedimiento general:

-Se establece una probabilidad de mutación p_m, la que ayudará a determinar la cantidad promedio de individuos que serán mutados: p_m x tamaño población.

-Cada gene dentro de un cromosoma tiene la misma probabilidad de ser mutado, de tal manera que para cada cromosoma de la población actual (después del cruce) y para cada gene dentro de cada cromosoma se genera un número aleatorio a en el rango de [0..1]. Si a<pm se muta el gene correspondiente.

-Se tiene entonces que si para un padre c (cromosoma) el gene x_k es seleccionado, el cromosoma resultante $c' = (x_1, x_2, ..., x'_k...x_n)$ donde:

La función Δ(t,y) regresa un valor en el rango [0,y] tal que la probabilidad de que Δ(t,y) esté cerca de cero incremente así como t lo haga, siendo t el número de la generación correspondiente. Tal propiedad provoca que el operador busque en el espacio uniformemente al inicio (cuando t es pequeño), y muy localmente en los estados finales. Se empleará la siguiente expresión:

$$\Delta(t,y) = y.\boldsymbol{a}.\ (1 - t/T)^{nu}, \qquad (16)$$

en donde **a** es un número aleatorio entre [0..1], **T** es el número máximo de generaciones y es un parámetro del sistema que determina el grado de no uniformidad.

Ejemplos numéricos

Con la finalidad de mostrar el desempeño del algoritmo genético diseñado, se han seleccionado dos casos de aplicación de optimización con objetivos múltiples, el primero es un ejemplo clásico de optimización con varios objetivos y restricciones lineales, el segundo corresponde a un problema complejo de optimización con varios objetivos representados con funciones no lineales y una restricción no lineal. Se han tomado ambos casos debido a que el primero representa un caso típico de optimización para la toma de decisiones y el segundo por tratarse de un caso numérico interesante por contener funciones objetivos multimodales en términos de la función trigonométrica seno, lo que genera la consecuencia de que los métodos tradicionales no pueden hacer casi nada en tal problema.

Para la implementación del algoritmo genético se seleccionó el lenguaje de programación C++ y ejecutados en un Pentium III, 500 Mhz.

RESULTADOS Y DISCUSIÓN

Planeación de Inversiones Bancarias

Los resultados obtenidos con el AG diseñado para éste problema se muestran en la tabla Nº1.

TABLA Nº1. Ejemplo bancario

Meta # 1	Meta #2	Meta # 3	(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)	Pm
18.347	0.931	6.871	24.896,13.441,13.36,13.625, 47.825,43.78,14.232,79.4	0.4
18.43	0.937	7.107	24.695,13.017,12.65,13.217, 43.812,43.66,13.887,84.585	0.4
18.128	0.915	6.806	24.462,16.966,14.595,13.484, 44.694,38.293,13.996,83.828	0.3
18.024	0.907	6.636	24.3,16.768,13.625,13.215, 47.712,43.258,14.175,75.285	0.3
18.418	0.923	6.980	24.71,16.658,13.106,13.624, 42.888,50.773,13.293,75.536	0.3
*18.5	0.928	7.000	24.2,16.03,12.5,12.5,44.77, 52.5,12.5,75.0	0.3
18.359	0.894	6.821	24.792,16.454,13.367,12.562, 47.487,48.115,12.995,75.305	0.2

Las metas deseadas eran obtener un beneficio ≥ 18.5 millones de $, relación de capital ≤ 0.8, relación de riesgo ≤ 7.0. Los resultados obtenidos son muy satisfactorios desde el punto de vista de su calidad y como orientación en la toma de decisiones cuando se desean alcanzar varios objetivos que suelen estar en conflictos. Se cree que modificando la función de aptitud incluyendo un procedimiento que ordene a las soluciones (cromosomas) por un grado basado en la evaluación de la función propuesta, las soluciones puedan mejorar ya que se podría hacer una mejor selección. La solución marcada en la tabla Nº1 con asterisco pudiese ser considerada la mejor en términos de cumplir dos de las metas.

Ejemplo Numérico No Lineal

Min: $Z(X) = 0.325\ u_1^- + 0.275\ u_2^- + 0.225\ u_3^- + 0.175\ u_4^-$ (17)

Sujeto a:

$z_1(x) - 18.0 = u_1^+ - u_1^-$ (18)

$z_2(x) - 15.0 = u_2^+ - u_2^-$ (19)

$z_3(x) - 10.0 = u_3^+ - u_3^-$

$z_4(x) - 0.0 = u_4^+ - u_4^-$

$G_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 \leq 100$

. x_i , u_i^-, $u_i^+ \geq 0$

El problema ha sido resuelto con el algoritmo genético diseñado y algunas modificaciones que tienen que ver con la forma en la cual son suministradas al programa las funciones objetivos y las restricciones. El tamaño de la población

permaneció constante en 50 individuos, se corrió para 5000 generaciones cada vez y tres veces para cada probabilidad de mutación; la probabilidad de cruce se mantuvo fija en 0.75. Los resultados pueden ser apreciados en la tabla 2.

TABLA Nº2. Resultados aplicación numérica

Meta # 1	Meta # 2	Meta # 3	Meta # 4	(x_1, x_2, x_3, x_4, x_5)
*18.328	16.844	11.835	7.112	(-1.620334,-6.757785,-4.727351,5.067248,1.664905)
17.642	16.675	11.643	4.840	(-2.824226, -3.992475, 6.841994, 5.073146, -1.408045)
*19.420	15.492	11.953	5.143	(1.551950, 4.199315, -6.810045, 3.540358, -4.149764)
19.167	15.426	9.016	5.640	(-1.917195, 3.865549, 4.473853, 6.613726, 4.176275)
18.362	15.564	8.925	4.437	(1.184057, 4.262119, -4.629875, 6.713028, 2.873358)

Las soluciones obtenidas en general son buenas aproximaciones a lo que se intenta y ofrecen también un espectro de soluciones posibles que permitirán la selección adecuada a la aplicación práctica pertinente. Estas soluciones fueron obtenidas en un tiempo comprendido en el rango de 45 a 62 segundos, lo que se puede calificar como un buen desempeño a pesar de que puede ser mejorado. Variando la combinación de pesos otorgados a cada meta, se obtendrán resultados diferentes acordes a tales ponderaciones. Una corrida de 5000 iteraciones y ponderaciones iguales para cada objetivo proporcionó la solución óptima Pareto = (-1.787, 0.722, 6.784,6.852, 1.895) bastante mejor y acorde a la combinación equitativa de pesos.

Los resultados obtenidos muestran en general un buen desempeño del algoritmo proporcionando resultados acordes con el dominio del problema. El programa diseñado genera soluciones que son obtenidas en un tiempo que depende en cierta medida de la cantidad de variables, de la cantidad de restricciones y de los valores asignados a los parámetros del algoritmo genético. Los métodos tradicionales requieren, previo al uso de cualquier paquete de programación lineal, la conversión del modelo no lineal a un modelo lineal; este paso requiere invertir mayor tiempo, se obtiene una aproximación del modelo real y en algunas ocasiones no puede realizarse si alguna de las funciones objetivo no puede ser diferenciable. El algoritmo genético encuentra buenas soluciones, en un tiempo de computación aceptable, lo que pudiese ser considerado como una demostración de la validez de este enfoque para la resolución de este tipo de problemas.

REFERENCIAS

Charnes, A. y Cooper, W. (1961). *Management Models and Industrial Applications of Linear Programming*. New York: John Wiley & Sons.

Forrest, S. (1996). Genetic Algorithms, en *CTC Handbook of Computer Scieece and Engineering*, A.B. Tucker (Editor). Boca Ratón, FL: CRC Press.

Hwang, C. y Masud, A. (1979). *Multiple Objetive Decision Making: Methods and Applications*. Berlin: Springer.

Ignizio, J. (1976). *Goal Programming and Extensions*. Lexington, MA: Heath.

Joines, J. y Houck, C. (1994). On the Use of Non-stationary Penalty Functions To Solve Nonlinear Constrained Optimization Problems with GAs en *Proceedings of the First IEEE Conference on Evolutionary Computation*, 579-584.

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs*. New York: Springer.

Rardin, G. (1998). *Optimization in Operations Researchs*. New York: Prentice Hall.

Saber, H. y Ravindran, A. (1993). Nonlinear goal programming theory and practice: a survey. *Computers and Operations Research*, 20 (3), 275-291.

Weistroffer, H. (1983). An interactive goal programming method for nonlinear multiple criteria decision – making problems. *Computers and Operations Research*, 10 (4), 311-320.

Autora

Jackelin Loyo de Sardi. Profesora Titular (Jubilada Activa) a Dedicación Exclusiva, Departamento Facultad de Ciencia y Tecnología de la Universidad de Carabobo, Valencia Venezuela. Ingeniero Industrial (U.C.) y Doctora en Ingeniería Industrial de la Universidad Anahuac, México.

E-mail: jloyo@uc.edu.ve

Recibido: 31/08/2011 **Aceptado: 16/12/2011**