EPARTAMENTO DE ESTADISTICA
E INVESTIGACION OPERATIVA

UNIVERSIDAD DE SEVILLA

FACULTAD DE MATEMATICAS

METODOS DE MUESTREO PARA LA

OPTIMIZACION GLOBAL ENTERA

Visado en Sevilla a
20 de Septiembre de 1988

Fdo: Prof. Dr.Fco Ramón
Fernández GArcía

Memoria presentada por
Juan del Ojo Mesa, para
optar al grado de Doctor
en Ciencias Matemáticas.

Fdo: Juan del Ojo Mesa

CAPITULO 0 : INTRODUCCION

Existen multitud de problemas en las Ciencias Aplicadas para los cuales el modelo matemático que los representa -- adopta la forma :

$$P : \text{minimizar } f(x)$$

$$x \in S \subset \mathbb{R}^n$$

donde x es la variable de decisión, que consideraremos perteneciente al espacio euclideo n-dimensional $\mathbb{R}^n$, la cual - está sometida a una serie de restricciones. El conjunto de todos los valores de la variable que las verifican lo repre

sentaremos por S y lo llamaremos región factible. f es una función real n-dimensional denominada función objetivo.

En la literetura han surgido gran cantidad de procedimientos (1) para resolver P que facilitan tan solo una solución parcial, pues determinan un punto mínimo local, o sea, hallan un punto x_* de S tal que existe un entorno E del mismo de modo que

$$f(x_*) \leq f(x) \qquad \forall x \in S \cap E$$

En general existirán varios puntos de este tipo y sus valores funcionales pueden diferir sustancialmente.

El problema de diseñar algoritmos que determinen al punto mínimo local con menor valor funcional se denomina problema de optimización global , y consistirá por tanto en hallar un punto $x^* \in S$ tal que verifique

$$f(x^*) \leq f(x) \qquad \forall x \in S$$

Si suponemos que la función es continua y que S es compacto existirá tal punto y a su valor funcional f* lo denominaremos mínimo global. Obsérvese que el valor f* es único aunque pueden existir diversos puntos mínimos locales diferentes.

Es natural que al problema de buscar puntos mínimos locales se le encontrara una rápida solución, pues si por ejemplo la función es dos veces diferenciables con continuidad existe una condición necesaria y suficiente para verificar si un punto es mínimo local basada en el conocimiento de las dos primeras derivadas en dicho punto. Si el resultado de tal condición resulta negativo, la diferenciabilidad con continuidad asegura la existencia de un punto en una vecindad del anterior con un menor valor funcional. Por tanto se puede construir una sucesión de puntos convergente hacia un mínimo local.

Si la función posee tan solo un punto mínimo local será también global y por tanto los procedimientos clásicos son válidos en estos casos. Así ocurre con las funciones convexas o con la clase más amplia de funciones pseudo y cuasi-convexas (1).

Para una función arbitraria no existe hoy dia una solución satisfactoria de P. La explicación de esto radica en los dos hechos siguientes :

1.- No se conocen condiciones suficientes para saber si un mínimo local es global.
2.- Para cualquier función objetivo f diferenciable con continuidad, cualquier punto $\underline{x}$ y cualquier vecindad E de $\underline{x}$ existe una función g tal que f+g es igual a f en todos los puntos fuera de E,y el mínimo global de f+g se alcanza en $\underline{x}$ (2).

Por tanto, para cualquier punto $\underline{x}$ no se puede afirmar que es el mínimo global sin evaluar la función en al menos un punto de cada vecindad E de $\underline{x}$. Como ésta puede elegirse arbitrariamente paqueña, se sigue que cualquier método diseñado para resolver el problema de óptimización global requerirá un número ilimitado de pasos; es decir, con un número finito de iteraciones tan solo se obtendrá una solución aproximada.

De aquí que el problema de la optimización global se considerará resuelto si dado $\varepsilon > 0$ es posible determinar un elemento de alguno de los siguientes conjuntos :

$$A_x(\varepsilon) = \{ x \in S \ / \ \| x - x^* \| < \varepsilon \}$$

$$A_f(\varepsilon) = \{ x \in S \ / \ |f(x) - f(x^*)| < \varepsilon \}$$

$$A_\phi(\varepsilon) = \{ x \in S \ / \ \phi(f(x)) < \varepsilon \}$$

donde $\phi(t) = \dfrac{m(\{ y \in S \ / \ f(y) < t \})}{m(S)}$, siendo m() la medida de Lebegue n-dimensional.

Los tres conjuntos contienen al punto mínimo global. Una desventaja del primero es que pequeñas perturbaciones en los datos del problema pueden tener efectos mayores sobre la localización de x*. El tercer conjunto puede contener puntos cuyos valores funcionales sean considerablemente mayores que f*.

Los métodos o algoritmos desarrollados para encontrar un elemento de cualquiera de los conjuntos anteriores se dividen en dos grandes grupos, dependiendo en la incorporación o no de elementos de caracter probabilísticos.

Los algoritmos determinísticos intentan localizar un elemento de $A_f(\varepsilon)$ tras una búsqueda exhaustiva en S. Por lo antes expuesto son inevitables hipótesis adicionales sobre la función objetivo. La más habitual es la de considerar una función lipschitciana, pues así se tiene acotada la variabilidad de la función entre pares de puntos.

El método de Evtushenko (3) consiste en obtener un recubrimiento por bolas de S basándose en el conocimiento de la constante lipschitciana. Con esta misma hipótesis Shubert (4) propone un algoritmo en el cual aproxima la función por superficies planas.

Se han obtenido mejores resultados con otros procedimientos determinísticos pero al precio de imponer más hipótesis sobre la función o de no tener certeza con la aproximación obtenida. Así, el métood de las trayectorias de Branin (5) y Hardy (6) consiste en obtener una curva que pase por todos los puntos estacionarios de la función, la cual se obtiene mediante la integración numérica de una ecuación diferencial asociada.

Se pueden también resaltar , entre otros, el método de deflación de Goldstein y Price (7) basado en el desarrollo de Taylor de la función objetivo, y el método de Levy (8) que reduce el problema al de encontrar un cero de una función asociada.

Los métodos determinísticos son interesantes desde el punto de vista teórico pues proporcionan una solución del problema con la precisión que deseemos, pero en general no son aplicables en la práctica debido a sus dos grandes inconvenientes :

1.- Imponen severas restricciones a la función objetivo.
2.- El esfuerzo computacional que requieren estos métodos suele ser excesivo, generalmente crece exponencialmente con la dimensión n del problema.

Mejores resultados se obtienen mediante los procedimientos estocásticos, aleatorios, probabilísticos o también llamados de Monte Carlo. Se caracterizan por incorporar elementos probabilísticos en su ejecución; mediante ellos se sacrifica el conocimiento de la precisión de la aproximación obtenida pero se obtienen estimaciones muy aceptables con un menor esfuerzo computacional, y en general tan solo exigen la continuidad de la función objetivo. La bondad de la estimación del mínimo global que se obtiene puede controlarse en términos probabilísticos.

Existen básicamente dos formas de introducir el elemento aleatorio en el algoritmo : mediante la elección de una dirección de búsqueda aleatoria o mediante al obtención de una muestra probabilística en la región S.

Los algoritmos aleatorios de búsqueda global se basan en la primera idea. Son secuenciales por naturaleza pues la estructura básica de estos métodos viene dada por la siguiente fórmula iterativa :

$$y_k = x_k + b_k$$

$$x_{k+1} = \begin{cases} y_k & \text{si } f(y_k) < f(x_k) \\ x_k & \text{en otro caso} \end{cases}$$

donde b_k se obtiene de una distribución de probabilida u_k

definida sobre S. Se obtienen diferentes algoritmos eligiendo diversas distribuciones : lo natural es hacer depender u_k de los puntos previos $x_1,\ldots,x_k$ y de sus respectivos valores funcionales, para así obtener un algoritmo que va aprovechando la información que se obtiene en cada iteración. La elección de las distribuciones de probabilidad debe ser tal que se tenga asegurada la convergencia en probabilidad de la sucesión; es decir, que se verifique que :

$$\lim_{k \to \infty} P (x_k \in A) = 1$$

donde A es cualquiera de los tres conjuntos antes citados. En (9) se dan condiciones necesarias que han de verificarse para que se tenga asegurada dicha convergencia. Estos algoritmos suelen ser de convergencia lenta pero son muy fáciles de implementar en un ordenador y ocupando poca memoria (10), además son robustos por lo que son útiles cuando la dimensión del problema es grande.

Los algoritmos aleatorios que se basan en una muestra probabilística producen mejores resultados. La mayoría de ellos consisten en dos fases : fase global y fase local .

En la primera se intenta garantizar la convergencia en probabilidad y para ello se evalúa la función sobre un determinado número de puntos escogidos según una cierta distribución sobre S, que generalmente es la uniforme pues la información de partida es nula. La importancia de esta fase radica en el siguiente hecho : si m(S) es finita y A c S verifica $0 \leq m(A) \leq m(S)$ entonces la probabilidad de que al menos un punto pertenezca a A de entre N elegidos uniformemente sobre S viene dada por :

$$p(A,N) = 1 - \left(1 - \frac{m(A)}{m(S)} \right)^N$$

y por tanto

$$\lim_{N \to \infty} p(A,N) = 1$$

Durante la fase local los puntos obtenidos se manipu-- lan mediante un algoritmo local para obtener un candidato a la solución del problema; se pretende co ello mejorar la eficiencia del algoritmo. (Un algoritmo local es tal que partiendo de un punto inicial de S localiza un punto mínimo local. Dependiendo de las hipótesis que se hagan sobre la función existen en la literatura un gran número de tales algoritmos (1,11,12)).

El método más simple es el algoritmo de búsqueda aleatoria pura (13,14), el cual consta tan solo de la fase global, tomándose como estimación el punto con menor valor funcional de entre los N tomados. Si f es continua se tiene asegurada la convergencia casi-segura del algoritmo (15,16), pero la convergencia es muy lenta por lo que no es útil en la práctica, pero sí como base para otros algoritmos más sofisticados.

Una primera extensión consiste en utilizar un algoritmo local, así obtenemos el algoritmo de partida multiple (21) que aplica a cada punto obtenido en la muestra el algoritmo local. La estimación será el mínimo local obtenido con menor valor funcional. La convergencia de la fase global se sigue conservando pero posee una gran deficiencia : un mismo mínimo local puede obtenerse diversas veces. Para intentar solventar este problema, dado un algoritmo local, se define la región de atracción de un mínimo local como el conjunto de puntos de S tales que tomados como puntos iniciales del algoritmo local se obtiene dicho mínimo local. Basta pues obtener un punto en cada región de atracción y aplicarle el algoritmo a cada uno.

Los algoritmos que se basan en la idea anterior hacen uso de la técnicas de análisis cluster para determinar las regiones de atracción,y son los más eficientes hasta el momento. (El análisis cluster consta de una serie de técnicas estadísticas cuyo objetivo es dividir un conjunto de datos en subconjuntos de elementos "similares" (17).)

Se parte pues de una muestra uniforme sobre la región S y mediante las técnicas anteriores se crean grupos de puntos cercanos que correspondan a las distintas regiones de atracción, y de cada una de ellas se aplica el algoritmo local.

El primero en utilizar un algoritmo de esta naturaleza fué A.A. Törn (18,19), el cual construye aproximaciones de las regiones de atracción usando elipsoides. La principal desventaja estriba en que impone la forma de dichas regiones. Este incoveniente es solventado por G.T. Timmer en sus algoritmos al no imponer a priori forma alguna. Un estudio detallado de ellos junto con una comparación con otros algoritmos puede verse en (20), resultando que son los más eficientes hasta el momento.

Concluimos esta breve exposición sobre el problema de la optimización global con la idea de que no se ha resuelto satisfactoriamente, pero los algoritmos que ofrecen un mejor resultado son los de tipo estocásticos, y dentro de ellos los basados en las técnicas de análisis cluster (21).

En esta memoria vamos a imponer una restricción adicional al problema P : la variable de decisión tan solo tomará valores enteros.

Muchas veces un valor no entero de la variable no tiene una interpretación física, basta considerar problemas de contaje, secuenciación, de recubrimiento ,etc. (22)

El modelo matemático adoptará la forma :

$$P_e : \begin{array}{l} \text{minimizar} \quad f(x) \\ x \in S \cap Z^n \end{array}$$

donde Z^n es el conjunto de puntos del espacio euclideo $\mathbb{R}^n$ cuyas componentes toman valores enteros. La región factible $S \cap Z^n$ la llamaremos malla entera.

Pretendemos determinar algoritmos que nos localicen un punto mínimo global entero , o sea, que hallen un punto

$m^* \in S \cap Z^n$ tal que $f_e^* = f(m^*) \leq f(x)$ $\forall x \in S \cap Z^n$

Este es el problema de la optimización global entera. Obsérvese que asociado a P_e tendremos el modelo P de optimización global que se obtiene al eliminar la restricción, al cual llamaremos modelo o problema continuo.

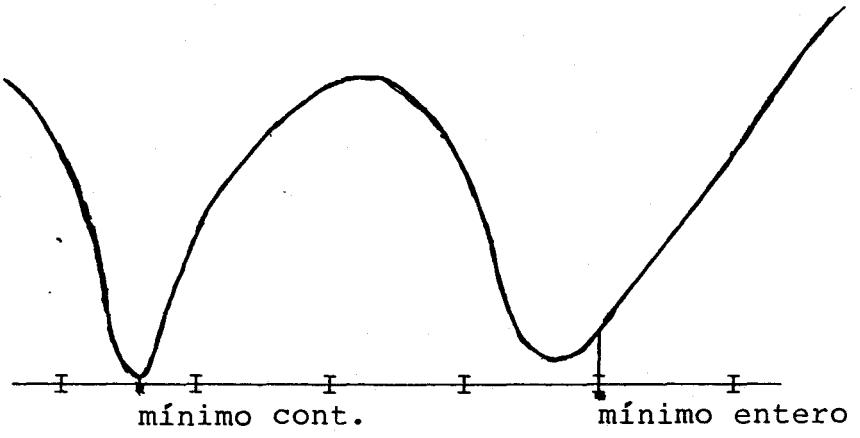
Aparentemente no parece que dicha restricción represente un serio problema, sino todo lo contrario : de una región factible de tipo continuo se pasa a otra discreta o numerable. Pero la conclusión no puede ser más erronea; si bien la región factible está mejor estructurada en la optimización global entera, en el sentido de pasar de un conjunto continuo a otro discreto, computacionalmente es más complejo como se ha venido viendo desde finales de 1940 en que se empezó a estudiar este problema. La causa principal es que la restricción adicional destruye las propiedades topológicas básicas de la región factible; así por ejemplo, se pierde la noción de continuidad : aún cuando la función objetivo sea continua nada puede afirmarse sobre los valores funcionales de puntos cercanos en la malla entera, aquellos pueden diferir sustancialmente.

Al igual que en el modelo continuo, se puede definir el concepto de punto mínimo local entero, la única diferencia es que para la definición de entorno se suele emplear la maxi-norma por ser la más natural :

$$\forall x \in Z^n \quad ,, \quad \|x\| = \max_{i=1,\ldots,n} |x_i|$$

El punto mínimo local entero con menor valor funcional nos proporcionará un punto mínimo global entero, que es la solución de P_e. (Cuando no empleemos la palabra punto y tan solo hablemos de mínimo nos referiremos al valor funcional; así los mínimos globales continuo y entero serán únicos)

En un primer momento cabría pensar que existiera alguna relación entre los puntos mínimos globales enteros y continuos, por ejemplo que al redondear al entero más próximo las coordenadas de uno de los segundos se obtendría uno de los primeros. Lo cierto es que no existe ningún tipo de relación entre ellos, como se puede observar en un sencillo ejemplo unidimensional :



Por tanto se han de determinar algoritmos específicos para la resolución del problema de optimización global entera.

No es restrictivo suponer que la malla entera está acotada, y por tanto que el número de puntos a investigar sea finito. Como consecuencia la forma primera y más simple de resolver P_e será mediante una total enumeración de los mismos. Esto será factible cuando ese número de puntos sea tratable por un ordenador en un tiempo prudencial (23); pero cuando esto no se cumpla no nos servirá el método. Habrá pues que idear la forma de eliminar puntos de la malla sin tener que evaluar la función sobre ellos. Esto se consigue en el caso particular de una función objetivo lineal y una malla dada por un sistema de inecuaciones lineales. Esta es sin duda la situación que más se ha estudiado en la literatura de la optimización global entera dándose además la particularidad de que el mínimo local es global.

Aquí vemos de nuevo como la retricción impuesta a P_e complica enormemente el problema : en el caso lineal el proble

ma continuo asociado se resuelve fácilmente a través del método del simplex (25) que se basa en la propiedad de que en tales condiciones la región factible es convexa y el mínimo global se alcanza en un extremo de la misma; sin embargo en el caso entero se pierde tal propiedad. Los algoritmos ideados son por tanto más complejos que el método del simplex. Destacamos entre ellos el método de branch & bound y el de los planos de corte (22) ; en ambos métodos se trabaja con el problema continuo (pues se sabe resolver) modificándolo convenientemente para que el mínimo global continuo sea el entero que buscamos. Aunque se tiene garantizada la convergencia en un número finito de pasos el aspecto computacional de ambos métodos empeora a mediad que crece la dimensión del problema.

Dejando el caso lineal tan solo se conocen algoritmos para un reducido número de situaciones muy particulares (24).

El problema de optimización global entera es pues de gran complejidad y aún en situaciones muy particulares los algoritmos de tipo determinísticos que se han ideado no son efectivos computacionalmente.

En esta memoria trataremos el caso de una función objetivo arbitraria a la que tan solo exigiremos la continuidad, y daremos algoritmos de tipo estocásticos que resuelven el problema de la optimización global entera en los términos que se especificarán, que se caracterizan por su fácil implementación en un microordenador y por la bondad de las estimaciones que facilitan empleando para ello una pequeña fracción de muestreo.

Hemos dividido la memoria en cuatro capítulos.

En el capítulo I estudiamos los algoritmos secuenciales estocásticos. La finalidad de éstos es obtener una sucesión de puntos enteros factibles cuyos valores funcionales converjan al mínimo global entero. Para ello damos la estructura de un algoritmo genérico e imponiendo ciertas condiciones estudiamos la convergencia en probabilidad del mismo. Según aquellas obtenemos algoritmos locales o globales, cuya diferencia primordial radica en que los primeros centran su búsqueda en zonas reducidas de la región factible y los segundos lo hacen sobre zonas más extensas. Finalmente en este capítulo consideramos dos versiones del algoritmo genérico, a saber, el algoritmo uniforme y el algoritmo reticular.

En el capítulo II tratamos los algoritmos de asignación. Se parte de un algoritmo de búsqueda global entera T y se pretende mejorar su efectividad reduciendo la zona de búsqueda. Esto lo conseguimos introduciendo una partición sobre la región y unas probabilidades sobre cada una de las partes para a continuación determinar la forma óptima de asignar el número total de puntos a muestrear. Esto constituye lo que hemos denominado el algoritmo combinado.

Los resultados teóricos que obtenemos son válidos para el caso particular de tomar como T al algoritmo uniforme, pues para éste podemos determinar las probabilidades de localizar al mínimo global. No obstante, usando las asignaciones óptimas que se obtienen para el caso uniforme, el algoritmo combinado ofrece excelentes resultados como ponemos de manifiesto en el capítulo IV.

En el capítulo III consideramos la resolución del problema de optimización global entero desde otra perspectiva. Bajo la hipótesis de que se sabe resolver el problema continuo abordamos la resolución del problema entero. Para ello asociamos a éste un problema continuo cuyo mínimo global es pre

cisamente el mínimo global entero que buscamos. Esto se consigue añadiéndole a la función objetivo una función penalizadora que depende de un parámetro. Al incrementar este parámetro los mínimos globales continuos de esta nueva función tienden a acercarse al mínimo global entero de la función objetivo original.

En el último capítulo presentamos los resultados numéricos que hemos obtenido al comparar los algoritmos estudiados en los dos primeros capítulos de la memoria. Llegamos a la conclusión de que el algoritmo más eficiente es el algoritmo combinado. Con él se obtienen estimaciones muy próximas al mínimo global entero de todas las funciones test consideradas, tomando para ello una pequeña fracción de muestreo.

Finalmente presentamos la linea de investigación que seguiremos en el futuro basada en la mejora del tipo de muestreo que se realiza.

CAPITULO I : ALGORITMOS SECUENCIALES ESTOCASTICOS

I.1 INTRODUCCION

Consideremos el problema de optimización global entera definido por :

$$\begin{array}{l} \text{minimizar} \quad f(x) \\ x \in S \cap Z^n \end{array}$$

donde f es una función real n-dimensional continua sobre la región S del espacio euclideo n-dimensional $\mathbb{R}^n$.

Suponemos que el problema tiene solución; es decir :

$$\exists\, f_e^* \in \mathbb{R}\,,\ \exists\, n^* \in S \cap Z^n \ /\ f_e^* = f(n^*) \leq f(x)\ \ \forall x \in S \cap Z^n$$

f_e^* es el mínimo global entero de la función f sobre el conjunto S.

Definamos el conjunto $E^* = \{ x \in S \cap Z^n / f(x) = f_e^* \}$
Nuestro objetivo será determinar algún elemento del mismo.

En este capítulo estudiaremos unos algoritmos de naturaleza probabilística que denominaremos algoritmos secuenciales estocásticos, los cuales parten de un punto o vector semilla y a partir de él generan, de una forma secuencial y aleatoria, un conjunto de vectores $\{x_k\}$ tal que la sucesión de los valores funcionales de dichos vectores es monótona decreciente, y además se tenga garantizada la convergencia en probabilidad de los algoritmos; o sea , que se verifique :

$$\lim_{k \to \infty} P(x_k \in E^*) = 1$$

donde $P(x_k \in E^*)$ es la probabilidad de que en la iteración k-ésima el vector x_k generado pertenezca a E^*.

I.2 ALGORITMO GENERICO

Formalicemos lo dicho hasta ahora considerando el siguiente algoritmo secuencial :

PASO 0 : Tomar $x_0 \in S \cap Z^n = \underline{S}$. Hacer $k=0$.

PASO 1 : Generar b_k del espacio muestral $(Z^n, \mathcal{P}(Z^n), m_k)$

PASO 2 : Hacer $x_{k+1} = H(x_k, b_k)$. Elegir m_{k+1}, hacer $k=k+1$ y volver al paso 1.

donde las m_k son medidas de probabilidad sobre Z^n , y $H : \underline{S} \times Z^n \longrightarrow \underline{S}$ es una aplicación que verifica la siguiente condición :

(A1) $f(H(x,b)) \leq f(x)$, y si $b \in \underline{S}$ entonces además $f(H(x,b)) \leq f(b)$

Esta condición no hace más que exigir el caracter decreciente de los valores funcionales que se van obteniendo. La iteración k-ésima del algoritmo consiste en fijar k y llevar a cabo los pasos 1 y 2.

A continuación interesará estudiar las condiciones bajo las cuales se tendrá garantizada la convergencia en probabilidad del algoritmo genérico. Evidentemente dichas condiciones irán referidas a la sucesión de medidas de probabilidades $\{ m_k \}$. Entendemos estas medidas como probabilidades condicionadas a los puntos $x_0, x_1, \ldots, x_{k-1}$ generados anteriormente.

I.3 CONVERGENCIA

Parece obvio que la convergencia en probabilidad se podría obtener si en cualquier iteración todo subconjunto no vacio se $\underline{S}$ tuviera probabilidad positiva; o sea :

$$\forall A \subset \underline{S} \quad ,, \; m_k(A) = P(b_k \in A) > 0$$

y basta tener en cuenta la condición (A1). Expresemos esta idea formalmente mediante la siguiente definición :

Definición I.3.1

Se dirá que el algoritmo genérico es global si se verifica la siguiente condición:

(A2) $\quad \forall A \subset \underline{S}$, $\text{card}(A) > 0 \quad ,, \quad \prod_{k=0}^{\infty} (1 - m_k(A)) = 0$

Es decir, mediante un algoritmo global la probabilidad de no muestrear indefinidamente en cualquier subconjunto no vacio de $\underline{S}$ es nula.

A continuación veremos algunas condiciones que aseguren el caracter global de un algoritmo.

Proposición I.3.1

Si existe $\varepsilon > 0$ tal que $m_k(A) \geq \varepsilon \quad \forall k$, entonces $\prod_{k=0}^{\infty} (1 - m_k(A)) = 0$.

-D-

$$m_k(A) \geq \varepsilon \Longrightarrow 1 - m_k(A) \leq 1 - \varepsilon \Rightarrow \prod_{k=0}^{\infty} (1 - m_k(A)) \leq (1 - \varepsilon)^m$$

$$\prod_{k=0}^{\infty}(1 - m_k(A)) = \lim_m \prod_{k=0}^{m}(1 - m_k(A)) \leq \lim_m (1 - \varepsilon)^m = 0.$$

c.q.d.

Como consecuencia inmediata se obtienen los dos siguientes resultados :

Corolario I.3.1

Si $m_k(A) = m(A) \quad \forall k \quad$ y $m(A) > 0$ entonces $\prod_{k=0}^{\infty}(1-m_k(A))=0.$

Corolario I.3.2

Si existe $\lim_k m_k(A) \neq 0 \quad$ entonces $\prod_{k=0}^{\infty}(1 - m_k(A)) = 0.$

Todo algoritmo global converge en probabilidad, como lo asegura el siguiente teorema.

Teorema I.3.1

Supongamos que se satisfacen las condiciones (A1) y (A2). Sea $\{ x_k \}_{k=1}^{\infty}$ la sucesión aleatoria generada por el algoritmo genérico. Entonces :

$$\lim_{k \to \infty} \quad P(x_k \in E^*) = 1$$

-D-

Si $x_k \in \underline{S}-E^*$ entonces $b_i \in Z^n-\underline{E}$, $i=0,1,\ldots,k-1$ pues en caso contrario x_k estaría en $\underline{E}$ debido a (A1).

Por tanto :

$$P(x_k \in \underline{S}-E^*) \leq P(b_i \in Z^n-\underline{E}^*, \ i=0,1,\ldots,k-1) =$$

$$= \prod_{i=0}^{k-1} P(b_i \in Z^n-E^*) = \prod_{i=0}^{k-1}(1-P(b_i \in E^*)) = \prod_{i=0}^{k-1}(1-m_i(E^*)).$$

O sea,

$$1 \geq P(x_k \in E^*) = 1-P(x_k \in \underline{S}-E^*) \geq 1-\prod_{i=0}^{k-1}(1-m_i(E^*)).$$

Por tanto,

$$1 \geq \lim_k P(x_k \in E^*) \geq 1-\lim_k \prod_{i=0}^{k-1}(1-m_i(E)).$$

Concluimos pues que

$$\lim_{k\to\infty} P(x_k \in E^*) = 1$$

c.q.d.

Nótese que para demostrar el teorema sólo es necesario que se verifique la condición (A2) para el conjunto $A \equiv E^*$, pero en la práctica este conjunto no nos será conocido, y de ahí que se exija para cualquier subconjunto.

Un algoritmo global exige por tanto que los posibles puntos a generar se repartan sobre todo $\underline{S}$. Sería interesante relajar esta condición. Para ello damos la siguiente definición :

Definición I.3.2

Diremos que el algoritmo genérico es local si el conjunto soporte M_k de m_k consta de un número finito de puntos y además se verifica que:

$\text{card}(\underline{S} \cap M_k) < \text{card}(\underline{S})$, $\forall k$ excepto quizás un número finito de ellos.

Por consiguiente el muestreo no se extiende a todo $\underline{S}$, pero evidentemente la condición (A2) no se tiene porqué verificar para un algoritmo generico local, y por tanto no se tendrá asegurada la convergencia en probabilidad. Damos a continuación otra condición con la cual si se tendrá.

(A3) $\forall x_0 \in \underline{S}$, $\exists \gamma > 0, \exists \beta \in (0,1]$ tales que

$$m_k(d(H(x,b_k),E^*) < d(x,E^*) - \gamma \text{ ó } H(x,b_k) \in E^*) \geq \beta$$

$$\forall k, \ \forall x \in L_0 = \{ x \in \underline{S} / \ f(x) \leq f(x_0) \}$$

donde d(x,A) es la distancia del vector x al conjunto A.

Partiendo de un vector x_0 es evidente que el conjunto L_0 contiene al óptimo que buscamos. La condición anterior exige que en cualquier iteración, cualquiera que sea el vector x_k de L_0, la probabilidad de que el siguiente punto generado por el algoritmo esté en E* o a una distancia de E* menor que la que hay desde x_k, debe ser estrictamente positiva. Es decir, en cada nueva iteración, estamos más cerca de E* o entramos en él, con probabilidad no nula.

Con esta nueva condición el algoritmo genérico local converge en probabilidad :

Teorema I.3.2

Supongamos que para todo x_0 de $\underline{S}$ el conjunto L_0 es finito, y que se satisfacen las condiciones (A1) y (A3).

Si $\{x_k\}_{k=0}^{\infty}$ es una sucesión generada por el algoritmo genérico , entonces se verifica que :

$$\lim_{k \to \infty} P(x_k \in E^*) = 1$$

-D-

Sea x_0 el punto generado en el paso 0.

Como L_0 es finito existirá un número natural p tal que

$$d(x,y) < p \cdot \gamma \ , \ \forall x,y \in L_0$$

Consideremos los siguientes sucesos :

$$A_i = \{ d(H(x_i,b_i),E^*) < d(x_i,E^*) - \gamma \}$$

$$B_i = \{ H(x_i,b_i) \in E^* \} \qquad i=0,1,\ldots,p-1$$

Como se verifica (A3) tendremos que $P(A_i \cup B_i) \geq \beta$

Nótese que

$$\bigcap_{k=0}^{p-1} A_k \subset B_{p-1} = \{ x_p \in E^* \}$$

pues la ocurrencia de la intersección implica las desigualdades siguientes:

$$d(x_1,E^*) < d(x_0,E^*) -$$

$$d(x_2,E^*) < d(x_1,E^*) -$$

$$\ldots\ldots\ldots\ldots\ldots\ldots$$

$$d(x_p,E^*) < d(x_{p-1},E^*) -$$

y sumándolas obtendremos que

$$d(x_p,E^*) < d(x_0,E^*) - \gamma \cdot p \leq 0$$

El segundo miembro de esta desigualdad es negativo pues sabemos que

$$d(x,y) < p \cdot \gamma \quad \forall\, x,y \in L_0$$

y por tanto

$$d(x_p,E^*) = 0 \longrightarrow x_p \in E^* .$$

Como además, por (A1), si se verificase algún $B_k, k=0,\ldots,p-1$ se tendría que $x_p \in E^*$, deducimos que

$$\bigcap_{k=0}^{p-1} (A_k \cup B_k) \subset \{ x_p \in E^* \}$$

Luego

$$P(x_p \in E^*) \geq P(\bigcap_{k=0}^{p-1} (A_k \cup B_k)) = \prod_{k=0}^{p-1} P(A_k \cup B_k) \geq \beta^p$$

y por tanto,

$$P(x_p \notin E^*) \leq 1 - \beta^p$$

Análogamente se verificaría que,

$$A_p \cap A_{p+1} \cap \ldots \cap A_{2p-1} \subset \{x_{2p} \in E^*\} \Rightarrow \bigcap_{k=p}^{2p-1} (A_k \cup B_k) \subset \{x_{2p} \in E^*\}$$

$$\Rightarrow \quad P(x_{2p} \notin E^*) \leq 1 - \beta^p$$

En general probariamos que,

$$P(x_{kp} \notin E^*) \leq 1 - \beta^p \ , \ k=1,2,\ldots$$

Como $\{x_{kp} \notin E^*\} \subset \{x_p \notin E^*, x_{2p} \notin E^*, \ldots, x_{kp} \notin E^*\}$ por (A1), tendremos que,

$$P(x_{kp} \notin E^*) \leq \prod_{i=1}^{k} P(x_{ip} \notin E^*) \leq (1-\beta^p)^k$$

y por tanto,

$$P(x_{kp} \in E^*) \geq (1 - \beta^p)^k \ , \ k = 1,2,\ldots$$

Finalmente, por (A1), se verifica que

$$\{x_{kp+j} \in E^*\} \supset \{x_{kp} \in E^*\} \ , \ j=0,1,\ldots,p-1$$

y,por consiguiente,

$$P(x_{kp+j} \in E^*) \geq P(x_{kp} \in E^*) \geq 1 - (1 - \beta^p)^k \xrightarrow[k \to \infty]{} 1$$

Luego $\lim_{k \to \infty} P(x_k \in E^*) = 1$

c.q.d.

Es de esperar que un algoritmo local posea una mejor tasa de convergencia que uno global, pues aquél centra su búsqueda en una zona reducida de $\underline{S}$, mientras que el global debe muestrear en grandes zonas. No obstante, uno local será mucho más susceptible de ser "atrapado" por un mínimo local en vez de por el mínimo global. Debe tenerse presente que la convergencia en probabilidad no nos garantiza que toda realización del algoritmo conlleve al mínimo global.

I.4 APLICACION

A continuación vamos a considerar dos versiones particulares del algoritmo genérico. En ambas tomaremos la siguiente definición de la aplicación H :

$$H(x,b) = \begin{cases} x & \text{si } f(x) \leq f(b) \\ b & \text{en otro caso} \end{cases}$$

Es obvio que se verifica la condición (A1).

VERSION I : ALGORITMO UNIFORME

Tómese el algoritmo genérico con $m_k = m$ para todo k, siendo m la distribución uniforme discreta sobre $\underline{S}$.

Como consecuencia del corolario I.3.1 se verifica la condición (A2), y por tanto el algoritmo uniforme es global, garantizándose pues la convergencia en probabilidad.

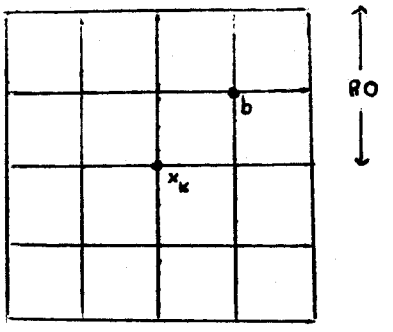
Obsérvese que realizar k iteraciones con este algoritmo equivale a obtener k puntos de forma uniforme sobre $\underline{S}$, y tomar como estimación el de menor valor funcional.

VERSION II : ALGORITMO RETICULAR

PASO 0 : Hacer $k = 0$, $\tilde{E}=\tilde{F}=0$; tomar $x_k \in \underline{S}$.

PASO 1 : Fijar los valores de las variables RO,PAS,ET,FT y R ($R \gg RO$), todos ellos enteros.

PASO 2 : Generar b según una ley uniforme en el rectángulo discreto de centro x_k y lados de amplitud $2 \cdot RO$.



PASO 3 : Tomar $x_{k+1} = H(x_k, b)$. Si $f(b) < f(x_k)$ entonces hacer $\tilde{E} = \tilde{E}+1$, $\tilde{F} = 0$. En otro caso $\tilde{F}=\tilde{F}+1$, $\tilde{E}=0$.

PASO 4 : Tomar

$$RO = \begin{cases} RO+PAS \text{ , si } \tilde{E} \geq ET. \text{ Si } RO>R \text{ tomar } RO=R. \\ RO-PAS \text{ , si } \tilde{F} \geq FT. \text{ Si } RO \leq 0 \text{ tomar } RO=1. \\ RO \text{ , en otro caso.} \end{cases}$$

PASO 5 : Tomar $k = k+1$ e ir al paso 2.

En cada iteración se muestrea en el cuadrado de lado de amplitud 2RO centrado en el punto obtenido en la iteración anterior. Diremos que se ha obtenido un éxito si el punto generado b disminuye el valor funcional, y en otro caso se habrá obtenido un fracaso. Cuando el número de exitos obtenido de forma ininterrumpida E alcanza un determinado valor ET se aumenta de tamaño el cuadrado de muestreo. Cuando el número de fracasos ininterrumpidos F es el que alcanza un determinado valor FT , disminuimos el tamaño. En cualquier otro caso dicho tamaño se mantiene. La cantidad constante por la que se aumenta o disminuye el tamaño viene dada por el parámetro PAS. El parámetro R nos da la máxima amplitud que puede alcanzar el cuadrado, la cual se limita para poder garantizar la convergencia del algoritmo como veremos a continuación. Actuando de esta manera se pretende trasladar el cuadrado sobre la zona que contiene al mínimo global entero.

Bajo la hipótesis de que el conjunto S es convexo se tiene la convergencia en probabilidad de este algoritmo de tipo local , pues la condición (A3) se verifica:

Dado $\gamma \in (0,1)$ se tiene que ,

$$m_k(d(H(x_k,b_k),E^*) < d(x_k,E^*) - \gamma \quad \text{ó} \quad H(x_k,b_k) \in E^*) \geq$$

$$m_k(d(H(x_k,b_k),E^*) < d(x_k,E^*) - \gamma) \geq \frac{1}{(2\cdot RO+1)^2} \geq \frac{1}{(2\cdot R+1)^2} > 0$$

pues siempre existirá al menos un punto en el cuadrado que diste una unidad menos de E* que desde x_k.

CAPITULO II : ALGORITMOS DE ASIGNACION

II.1 INTRODUCCION

En este capítulo vamos a considerar el problema de la optimización global desde el punto de vista del costo : a la hora de aplicar un algoritmo, uno de los puntos más importantes es el determinar un criterio de parada conveniente; lo ideal sería detener el proceso cuando la estimación actual esté "próxima" al mínimo global. Para ello son imprescindibles ciertas hipótesis sobre la función a minimizar para así obtener cotas sobre dicho mínimo.

Nosotros no estamos imponiendo condiciones a la función y por ello no podremos obtener un criterio de parada como el expuesto. Lo que si podemos hacer es limitar el número total de puntos a generar por el algoritmo, y nuestro criterio será detenerlo cuando se halla generado tal número de puntos.

Si denotamos por N ese número fijo de puntos, el algoritmo más simple, de tipo estocástico, que podríamos considerar sería el consistente en la obtención de una muestra de N puntos distribuidos de forma uniforme sobre la malla entera en cuestión, y tomar como estimación del mínimo global aquel punto con valor funcional mínimo. Aunque este procedimiento converge en probabilidad al crecer N indefinidamente, es necesario que éste tome un valor elevado para obtener estimaciones aceptables : Si la malla consta de M puntos, entonces la probabilidad de obtener al mínimo globla vendría dada por :

$$1 - \left(1 - \frac{1}{M} \right)^{N}$$

bajo la hipótesis simplificadora de que el mínimo se alcanza en un único punto. Si dicha probabilidad deseamos que valga $\nu \in (0,1]$, entonces el tamaño de muestra necesario sería :

$$N = 1 + INT \left(\frac{\ln(1-\nu)}{\ln(M-1) - \ln(M)} \right)$$

donde $INT(\cdot)$ denota la función parte entera.

La siguiente tabla muestra los valores de N para ciertos valores de ν y M dados:

ν \ M	500	1000	2000	3000	5000	10000
0.10	53	106	211	317	527	1054
0.40	256	511	1022	1553	2554	5104
0.60	458	916	1833	2749	4583	9163
0.70	602	1204	2408	3612	6020	12040
0.75	693	1386	2772	4159	6931	13863
0.99	2301	4603	9209	13814	23024	46050

Obsérvese que para valores de $\nu > 0.60$ se necesita tomar más puntos de los que hay en la malla, con lo cual el algoritmo dista mucho de ser efectivo.

Los puntos generados sobre la malla se toman de forma uniforme pues partimos de un conocimiento nulo a priori sobre la localización del mínimo. Si tuviésemos una información adicional sobre dicha localización y pudiésemos incorporarla al proceso de muestreo, obtendriamos una mejora del procedimiento.

La cuantificación de esa información la llevaremos a cabo mediante unas probabilidades a priori sobre la localización del mínimo.

II.1 ASIGNACIONES

Seguidamente presentamos la situación e hipótesis que mantendremos en todo el capítulo.

Partimos de una función $f : S \cap Z^n \longrightarrow \mathbb{R}$,donde S es un rectángulo n-dimensional del cual sólo nos interesará la malla entera formada por los puntos de S con coordenadas enteras.

Sea n* el único punto mínimo global entero en S.

Partimos de una subdivisión del rectángulo S en K rectángulos disjuntos $S_1, S_2, \ldots, S_K$ que efectúa una partición de S.

Sobre dicha partición suponemos conocida una distribución de probabilidad $p_1, \ldots, p_K$; $\sum_j p_j = 1$, de forma que p_j nos da la probabilidad de que el subrectángulo S_j contenga al mínimo n*; o sea,

$$p_j = P(n^* \in S_j) \quad ; j = 1,2,\ldots,K$$

Estas probabilidades constituyen la cuantificación de la información que disponemos sobre la localización de n*.

También supondremos dada una función de costo definida sobre la partición, de forma que dicha función nos mida el coste en que incurrimos al tomar n_j puntos en $S_j, j=1,2,\ldots,K$ el cual vendrá dado por $c(j,n_j)$.

Nuestro objetivo será determinar cuántos puntos tomar en cada subrectángulo S_j, de forma que la probabilidad de localizar al mínimo sea máxima, con la restricción de que el costo total no exceda de una cantidad C fijada por nosotros.

Para expresar formalmente lo antes expuesto damos las siguientes definiciones :

Definición II.2.1

Llamaremos asignación a una K-túpla de números enteros no negativos $A = (A_1, A_2, \ldots, A_K)$, de forma que A_j representa el número de puntos a tomar de forma uniforme en S_j mediante la asignación A.

Definición II.2.2

Denotaremos por $\mathcal{A}$ al conjunto de todas las posibles asignaciones.

Dado el sistema de probabilidades $\{p_j\}$ representaremos por $P(j,n_j)$ la probabilidad de localizar al mínimo tomando n_j puntos de forma aleatoria en S_j condicionada a que dicho mínimo se encuenra en S_j.

Por tanto, dada $A \in \mathcal{A}$ la probabilidad de localizar al mínimo con tal asignación vendrá dada por,

$$P(A) = \sum_{j=1}^{K} P(j,A_j) \cdot p_j$$

y el coste de tal asignación será,

$$C(A) = \sum_{j=1}^{K} C(j,A_j)$$

Nuestro problema será por tanto determinar $A^* \in \mathcal{A}$ tal que

$$P(A^*) = \max_{A \in \mathcal{A}} P(A)$$

sujeto a,

$$C(A) \leq C$$

A tal asignación A*, si existe, la denominaremos asignación C-óptima.

Antes de abordar la resolución del problema planteado hagamos unas puntualizaciones :

El valor $C(j,n_j)$ mide el costo en que incurrimos al tomar n_j puntos uniformemente en S_j. Dicho costo se podrá cuantificar de muy diversas formas, pero para continuar en la linea que expresamos al comienzo del capítulo, nos interesará la siguiente :

$$C(j,n_j) = n_j \quad , \quad \forall\, j$$

De esta forma la restricción $C(A) \leq C = N$ nos expresará que el número total de puntos a muestrar sobre S no debe exceder el valor prefijado N. Por tanto la asignación N-óptima nos indica la forma en que se deben distribuir los N puntos entre los K subrectángulos de forma que la probabilidad de localizar al mínimo sea máxima, dado el sistema de probabilidades p_j, $j=1,2,\ldots,K$.

Por otro lado, con la situación planteada, es evidente que,

$$P(j,n_j) = 1 - (1 - q_j)^{n_j}$$

siendo $q_j = 1 / \text{card}(S_j)$, donde $\text{card}(S_j)$ es el número de puntos con coordenadas enteras que contiene S_j.

II.3 ASIGNACIONES OPTIMAS PARA SU COSTE

Nuestro objetivo es determinar una asignación N-óptima, pero previamente, para poder llegar a ella, necesitamos de un nuevo tipo de asignaciones.

Definición II.3.1

Diremos que una asignación A^{**} es óptima para su costo si verifica que,

$$P(A^{**}) = \max_{\substack{A \in \mathcal{A} \\ C(A) \leq C(A^{**})}} P(A)$$

El siguiente resultado es elemental y nos da una condición que debe verificar una asignación para que sea óptima para su costo.

Proposición II.3.1

Dado $r \geq 0$ supogamos que existe una asignación $A_r^{**} \in \mathcal{A}$ que verifique :

$$P(A_r^{**}) - r \cdot C(A_r^{**}) \geq P(A) - r \cdot C(A) \quad , \quad \forall A \in \mathcal{A}$$

entonces dicha asignación A_r^{**} es óptima para su coste.

-D-

Sea $A \in \mathcal{A}$ que verifique $C(A) \leq C(A_r^{**})$; entonces se tiene

$$P(A_r^{**}) - P(A) \geq r \cdot (C(A_r^{**}) - C(A)) \geq 0$$

luego $P(A_r^{**}) \geq P(A)$

c.q.d.

Como consecuencia de este resultado y de la descomposición de $P(A)$ en una suma de K términos obtenemos el siguiente resultado :

Teorema II.3.1

Dado $r \geq 0$ definamos las siguientes funciones :

$$f_r^j(k) = p_j \cdot P(j,k) - r \cdot C(j,k) \quad ,k = 0,1,2,\ldots$$

Supongamos que $\forall j$, $\exists k_j^*$ tal que,

$$f_r^j(k_j^*) = \max_{k=0,1,2,\ldots} f_r^j(k)$$

y tomemos la asignación $A_r^{**} = (k_1^*, k_2^*, \ldots, k_K^*)$. Entonces dicha asignación es óptima para su costo.

-D-

Por hipótesis $f_r^j(k_j^*) \geq f_r^j(k)$, $\forall k$, por tanto $\forall A \in \mathcal{A}$ se verificará que , $f_r^j(k_j^*) \geq f_r^j(A_j)$, y sumando en j obtenemos,

$$\sum_j f_r^j(k_j^*) \geq \sum_j f_r^j(A_j)$$

con lo cual,

$$\sum_j (p_j \cdot P(j,k_j^*) - r \cdot C(j,k_j^*)) \geq \sum_j (p_j \cdot P(j,A_j) - r \cdot C(j,A_j)) \geq$$

$$\sum_j p_j \cdot P(j,k_j^*) - r \cdot \sum_j C(j,k_j^*) \geq \sum_j p_j \cdot P(j,A_j) - r \cdot \sum_j C(j,A_j)$$

obteniendo finalmente que,

$$P(A_r^{**}) - r \cdot C(A_r^{**}) \geq P(A) - r.C(A) \qquad \forall A \in \mathcal{A}$$

c.q.d.

El resultado obtenido es importante pues reemplaza el problema original con restricciones por K subproblemas unidimensionales sin restricciones.

Vamos a obtener la misma condición suficiente para que una asignación sea óptima para su coste, pero expresada de forma algo distinta y más operativa. Para ello definiremos unas nuevas funciones.

Sean

$$\underline{P}(j,k) = P(j,k) - P(j,k-1)$$

$$\underline{C}(j,k) = C(j,k) - C(j,k-1) \quad ;k=1,2,...$$

que representan respectivamente la probabilidad de que al tomar k puntos de forma uniforme en S_j se detecte el mínimo con el último extraido, y el costo de obtener el k-ésimo punto aleatorio. Evidentemente se verifican las relaciones siguientes :

$$P(j,k) = \sum_{i=1}^{k} \underline{P}(j,i)$$

$$C(j,k) = \sum_{i=1}^{k} \underline{C}(j,i) \quad ;k=1,2,...$$

Nótese que,

$$P(j,0) = C(j,0) \ , \ \forall\, j$$

Definamos las funciones siguientes:

$$g_r^j(k) = p_j \cdot \underline{P}(j,k) - r \cdot \underline{C}(j,k) \quad ;k=1,2,...$$

La condición anterior queda expresada de al siguiente forma :

Teorema II.3.2

Dado $r \geq 0$ supongamos que $\forall\, j, \exists\, k_j^*$ tal que,

$$g_r^j(k) \geq 0 \quad k=1,...,k_j^*$$

$$g_r^j(k) \leq 0 \quad \forall\, k > k_j^*$$

Tomemos $A_r^{**} = (k_1^*, k_2^*, \ldots, k_K^*)$; entonces dicha asignación es óptima para su coste.

-D-

Sea $A \in \mathcal{A}$. Supongamos que $k_j^* > A_j$, con j fijo. Entonces,

$$p_j \cdot (P(j,k_j^*) - P(j,A_j)) = \sum_{k=A_j+1}^{k_j^*} p_j \cdot \underline{P}(j,k) \geq r \cdot \sum_{k=A_j+1}^{k_j^*} \underline{C}(j,k) =$$

$$= r \cdot (C(j,k_j^*) - C(j,A_j)) \quad , \text{ pues } g_r^j(k) \geq 0 \text{ si } k \leq k_j^*.$$

Si $k_j^* \leq A_j$ también se llega al mismo resultado, pues :

$$p_j \cdot (P(j,k_j^*) - P(j,A_j)) = \sum_{k=1}^{k_j^*} p_j \cdot \underline{P}(j,k) - \sum_{k=1}^{A_j} p_j \cdot \underline{P}(j,k) =$$

$$= - \sum_{k=k_j^*+1}^{A_j} \underline{P}(j,k) \cdot p_j \geq - r \cdot \sum_{k=k_j^*+1}^{A_j} \underline{C}(j,k) = r \cdot \left(\sum_{k=1}^{k_j^*} \underline{C}(j,k) - \right.$$

$$\left. - \sum_{k=1}^{A_j} \underline{C}(j,k) \right) = r \cdot (C(j,k_j^*) - C(j,A_j)$$

siendo la desigualdad cierta pues por hipótesis $g_r^j(k) \leq 0$ si $k > k^*_j$.

Luego $\forall j$,, $p_j \cdot (P(j,k_j^*) - P(j,A_j)) \geq r \cdot (C(j,k^*_j) - C(j,A_j))$.

Sumando en j las K desigualdades anteriores obtenemos :

$$P(A_r^{**}) - r \cdot C(A_r^{**}) \geq P(A) - r \cdot C(A), \quad \forall A \in \mathcal{A}$$

c.q.d.

La condición expresada en el teorema anterior es suficiente para que una asignación sea óptima para su coste;con dos hipótesis adicionales también será necesaria.

Torema II.3.3

Supongamos que $\underline{C}(j,k) = \underline{P} > 0 \quad \forall j, \forall k$ y que $\underline{P}(j,\cdot)$ es decreciente para cada j como función de su segundo parámetro k. Entonces una condición necesaria y suficiente para que una asignación $A^{**} = (A_1^{**}, A_2^{**}, \ldots, A_K^{**})$ sea óptima para su costo es que exista $r \geq 0$ tal que:

$$g_r^j(k) \geq 0 \quad , k=1,\ldots,A_j^{**} \qquad \forall j \qquad (1)$$

$$g_r^j(k) \leq 0 \quad , \quad \forall k > A_j^{**} \qquad (2)$$

-D-

La condición es necesaria como consecuencia del teorema anterior. Mostremos que también es suficiente :

Partimos de $A^{**} \in \mathcal{A}$ que es óptima para su coste. Definamos el conjunto de índices

$$J = \left\{ j \;/\; A_j^{**} > 0 \right\}$$

y sea

$$r = \min_{j \in J} \; (1/\underline{P}) \cdot p_j \cdot \underline{P}(j, A_j^{**})$$

por definición $r \geq 0$.

Además $\forall j$,, $\underline{P} \cdot r \leq p_j \cdot \underline{P}(j, A_j^{**}) \leq p_j \cdot \underline{P}(j,k) \quad , \forall k \leq A_j^{**}$ por la monotonía de $\underline{P}(j,\cdot)$.

Pero $\underline{P} \cdot r = \underline{C}(j,k) \cdot r \quad , \forall k \quad$, y por tanto,

$$\underline{P}(j,k) \cdot p_j - r \cdot \underline{C}(j,k) = g_r^j(k) \geq 0 \quad \forall k \leq A_j^{**}$$

luego se verifica (1).

Supongamos que (2) no se verificase, entonces,

$$\exists\, j_0 \quad / \quad g_r^{j_0}(k_0) > 0 \quad \text{para algún } k_0 > A_{j_0}^{**}$$

o sea, se tendría que, $(1/\underline{P})\cdot p_{j_0}\cdot \underline{P}(j_0,k_0) > r$

Como $k_0 > A^{**}_{j_0}$ y ambos son naturales, tendremos $k_0 \geq 1+A^{**}_{j_0}$

y por tanto,

$$(1/\underline{P})\cdot p_{j_0}\cdot \underline{P}(j_0,1+A^{**}_{j_0}) \geq p_{j_0}\cdot(1/\underline{P})\cdot \underline{P}(j_0,k_0) > r \quad (3)$$

por la monotonía de $\underline{P}(j_0,\cdot)$.

Por definición de r, $\exists\, j_1 \;/\; (1/\underline{P})\cdot p_{j_1}\cdot \underline{P}(j_1,A^{**}_{j_1}) = r$

y teniendo en cuenta (3),

$$(1/\underline{P})\cdot p_{j_1}\cdot \underline{P}(j_1,A^{**}_{j_1}) < (1/\underline{P})\cdot p_{j_0}\cdot \underline{P}(j_0,1+A^{**}_{j_0}) \quad (4)$$

Definamos la asignación $\underline{A} = (\underline{A}_1,\underline{A}_2,\ldots,\underline{A}_K)$ dada por,

$$\underline{A}_j = \begin{cases} A^{**}_{j_0} + 1 & \text{si } j = j_0 \\ A^{**}_{j_1} - 1 & \text{si } j = j_1 \\ A^{**}_j & \text{si } j \neq j_0, j_1 \end{cases}$$

Es evidente que $C(\underline{A}) = C(A^{**})$ pues $C(A) = \underline{P}\cdot \sum A_j$ y por construcción $\sum A^{**}_j = \sum \underline{A}_j$

Como $P(A) = \sum p_j\cdot P(j,A_j) = \sum_j p_j\cdot \sum_{i=1}^{A_j} \underline{P}(j,i)$ tendremos,

$$P(\underline{A}) - P(A^{**}) = p_{j_0}\cdot \underline{P}(j_0,1 + A^{**}_{j_0}) - p_{j_1}\cdot \underline{P}(j_1,A^{**}_{j_1})$$

que es positivo debido a (4).

Luego $P(\underline{A}) > P(A^{**})$ y esto es contradictorio pues A^{**} es una asignación óptima para su costo.

c.q.d.

Como consecuencia de este teorema se tiene que las asignaciones óptimas para su costo se obtienen analizando las funciones $g_r^j(\cdot)$ para todos los valores posibles del parámetro r.

En nuestra situación particular vamos a estudiar para qué valores de r podemos obtener estas asignaciones óptimas asi como la expresión analítica de las mismas.

Recordemos que la situación que nos interesa es aquella en la cual,

$$C(j,k) = k \quad \forall j, \ \forall \ k$$

y por tanto,

$$\underline{C}(j,k) = 1 \quad \forall j, \quad \forall k$$

Vimos también que,

$$P(j,k) = 1 - (1 - q_j)^k$$

y, por tanto,

$$\underline{P}(j,k) = q_j \cdot (1 - q_j)^{k-1}$$

Obsérvese que $\underline{P}(j,\cdot)$ es monótona decreciente y que $\underline{C}(j,k)$ es constante. Estamos pues en condiciones de aplicar el teorema II.3.3 y obtener las asignaciones óptimas para su costo:

Como

$$g_r^j(k) = p_j \cdot q_j \cdot (1 - q_j)^{k-1} - r$$

es fácil deducir que,

$$g_r^j(k) \geq 0 \quad \text{sii} \quad k \leq 1 + \frac{\ln(r) - \ln(p_j \cdot q_j)}{\ln(1 - q_j)}$$

y, por tanto , la asignación $A_r^{**} = (A_1^{**}(r),\ldots,A_K^{**}(r))$,con

$$A_j^{**}(r) = \text{INT}\left(1 + \frac{\ln(r) - \ln(p_j \cdot q_j)}{\ln(1 - q_j)}\right)$$

es óptima para su coste, el cual viene dado por,

$$C(A_r^{**}) = \sum_{j=1}^{K} A_j^{**}(r) = \sum_{j=1}^{K} \text{INT}\left(1 + \frac{\ln(r) - \ln(p_j \cdot q_j)}{\ln(1 - q_j)}\right)$$

No todo valor de r será válido, pues se ha de cumplir que,

$$A_j^{**}(r) \geq 0 \qquad \forall\, j$$

y esto equivale, como es fácil de comprobar, a que se verifique la condición,

$$0 \leq r \leq \min_{j=1}^{K} \left(\frac{p_j \cdot q_j}{1 - q_j}\right)$$

Obsérvese que $A_j^{**}(r)$ es una función monótona decreciente en r, de forma que al acercarse r a cero por la derecha crece indefinidamente; y por tanto $C(A_r^{**})$ es también decreciente en r, verificándose que,

$$\lim_{r \longrightarrow 0^+} C(A^{**}) = +\infty$$

Dado N natural, nuestro problema consiste en determinar una asignación A** óptima para su coste, y de forma que éste

sea precisamente N, pues así habremos determinado la forma óptima de repartir los N puntos entre los K subrectángulos y tal que la probabilidadde detectar al mínimo sea máxima.

Tal asignación existe, pues tan sólo existen un número finito de posibles asignaciones con costo igual a N, y por tanto al menos una de ellas será de máxima probabilidad.

Teniendo en cuenta el teorema II.3.3, sabemos que tal asignación será de la forma A_r^{**} , para un cierto valor de r, el cual se habrá de determinar resolviendo la ecuación siguiente :

$$C(A_r^{**}) = N$$

que como hemos indicado antes posee solución en r. La única dificultad radica en que dicha ecuación no es fácil de resolver; téngase en cuenta la expresión del primer miembro. No obstante podriamos obtener una aproximación del valor de r teniendo en cuenta las propiedades que hemos resaltado sobre $C(A_r^{**})$ y mediante el siguiente algoritmo:

Paso 0 : Tomar $r < \min_j \dfrac{p_j \cdot q_j}{1 - q_j}$. Fijar $\Delta r \simeq 0$

Paso 1 : Determinar A_r^{**} mediante la fórmula obtenida.

Paso 2 : Si $C(A_r^{**}) < N$, hacer $r = r - \Delta r$. Ir al paso 1.

Si $C(A_r^{**}) > N$, hacer $r = r + \Delta r$. Ir al paso 1.

Si $C(A_r^{**}) \simeq N$, entonces FIN.

En el apartado siguiente daremos sin embargo la expresión explícita de dicha asignación.

II.4 ASIGNACIONES OPTIMAS DE COSTO N

En este apartado vamos a construir una determinada asignación A^* de costo N, y demostraremos que es óptima para su costo viendo que $A^* = A^{**}_r$ para un cierto valor de r y aplicaremos el teorema II.3.3.

Dado N natural fijo, consideremos la asignación A^*_N que se obtiene de la siguiente forma :

PASO 0 : Hacer $A(j,0) = 0$; $j=1,2,\ldots,K$

Para $i = 1,2,\ldots,N$ efectuar,

PASO i : Calcular

$$\max_{j=1}^{K} p_j \cdot \underline{P}(j,A(j,i-1)+1) = p_{j_i} \cdot \underline{P}(j_i,A(j_i,i-1)+1)$$

y definir,

$$A(j,i) = \begin{cases} A(j,i-1) & \text{si } j \neq j_i \\ A(j_i,i-1)+1 & \text{si } j = j_i \end{cases}$$

La asignación que consideraremos es :

$$A^*_N = (A^*_{N,1}, A^*_{N,2}, \ldots, A^*_{N,K}) \quad \text{donde} \quad A^*_{N,j} = A(j,N)$$

$$j = 1,2,\ldots,K$$

Por construcción, se verifica que :

$$A(j,0) \leq A(j,1) \leq \ldots \leq A(j,N) \quad ;J=1,2,\ldots,K$$

$$\left.\begin{array}{l} A(j_i,i) = A(j_i,i-1) + 1 \\ A(j,i) = A(j,i-1) \quad j \neq j_i \end{array}\right\} \quad i = 1,2,\ldots,N$$

El resultado al que queriamos llegar es el siguiente:

Teorema II.4.1

La asignación A_N^* es N-óptima.

-D-

Por construcción es evidente que $C(A_N^*) = N$.

Veamos que es óptima para su costo :

Definamos

$$r = \max_j \; p_j \cdot \underline{P}(j,A(j,N-1)+1) = p_{j_N} \cdot \underline{P}(j_N,A(j_N,N-1)+1) =$$

$$= p_{j_N} \cdot \underline{P}(j_N,A(j_N,N)) = p_{j_N} \cdot \underline{P}(j_N,A^*_{N,j_N})$$

El teorema quedará demostrado si demostramos que,

$$p_j \cdot \underline{P}(j,k) \geq r \qquad 1 \leq k \leq A^*_{N,j}$$

$$p_j \cdot \underline{P}(j,k) \leq r \qquad A^*_{N,j} < k < \infty$$

pues estas dos desigualdades no son más que,

$$g_r^j(k) \geq 0 \qquad 1 \leq k \leq A^*_{N,j}$$

$$g_r^j(k) \leq 0 \qquad A^*_{N,j} < k < \infty$$

y aplicariamos el teorema II.3.3.

En efecto :

a) Supongamos que $k > A^*_{N,j}$.

Tendriamos $k \geq A^*_{N,j} + 1$, y por tanto,

$\underline{P}(j,k) \leq \underline{P}(j,A^*_{N,j}+1)$, o sea, $p_j \cdot \underline{P}(j,k) \leq p_j \cdot \underline{P}(j,A^*_{N,j}+1)$

Pero $r \geq p_j \cdot \underline{P}(j,A(j,N-1)+1) \geq p_j \cdot \underline{P}(j,A(j,N)+1) =$

$= p_j \cdot \underline{P}(j,A^*_{N,j}+1)$

Luego,

$$\underline{P}(j,k) \cdot p_j \leq r \quad \text{si} \quad k > A^*_{N,j}$$

b) Supongamos que $k \leq A^*_{N,j}$, donde j es fijo. Para identificarlo hagamos $j = j_0$.
Como $k \leq A^*_{N,j_0}$, existirá una etapa i en la cual $j_0 = j_i$
y además $k = A(j_i,i)$. En dicha etapa se verificará que,

$$p_{j_i} \cdot \underline{P}(j_i,A(j_i,i-1)+1) \geq p_j \cdot \underline{P}(j,A(j,i-1)+1), \quad j=1,\ldots,K$$

o sea,

$$p_{j_i} \cdot \underline{P}(j_i,A(j_i,i) = p_{j_i} \cdot \underline{P}(j_i,k) \geq p_j \cdot \underline{P}(j,A(j,i-1)+1), \quad \forall\, j$$

Pero,

$$A(j,i-1)+1 \leq A(j,N-1)+1 \qquad \forall\, j$$

lo cual implica que,

$$\underline{P}(j,A(j,i-1)+1) \cdot p_j \geq \underline{P}(j,A(j,N-1)+1) \cdot p_j \quad , \quad \forall\, j$$

o sea ,

$$\forall j \;,\; p_j \cdot \underline{P}(j,A(j,N-1)+1) \leq p_j \cdot \underline{P}(j,A(j,i-1)+1) \leq$$

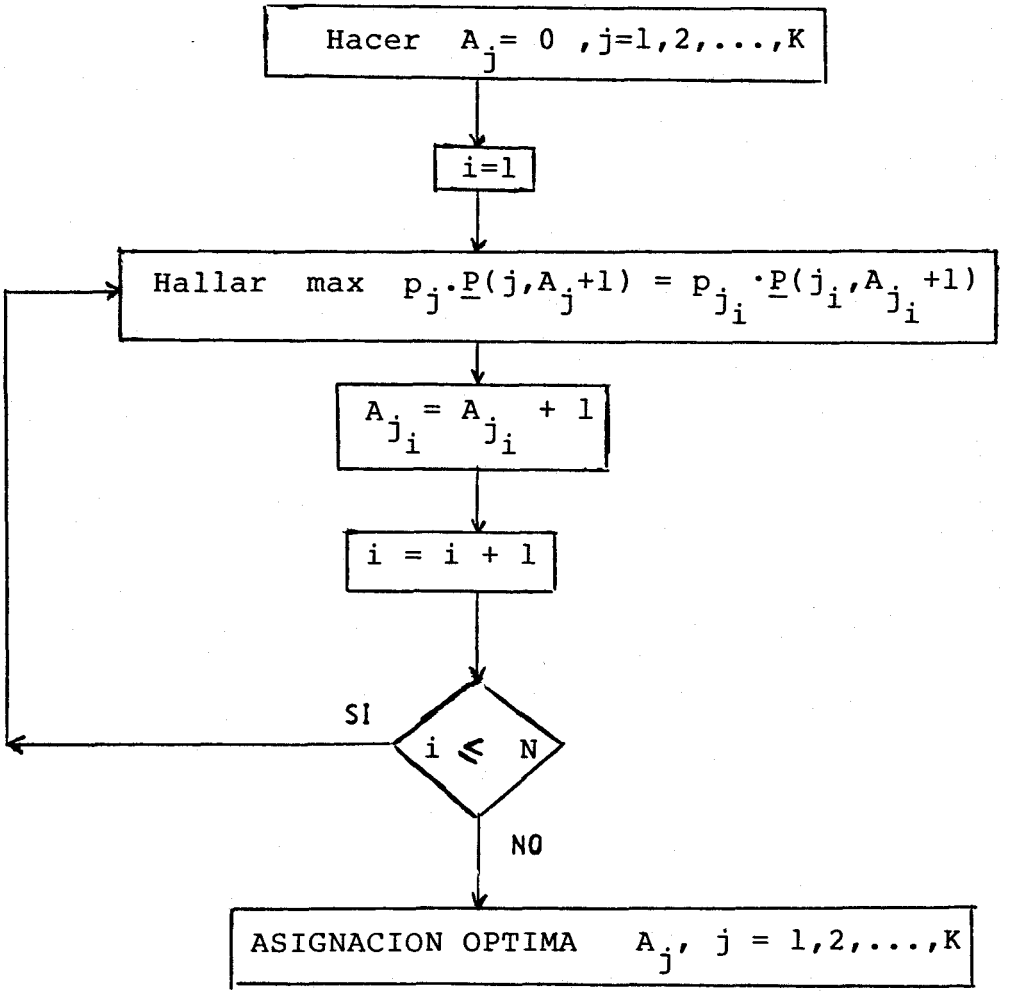
$\leq p_{j_i} \cdot \underline{P}(j_i, A(j_i, i))$

y tomando máximo en j obtenemos que,

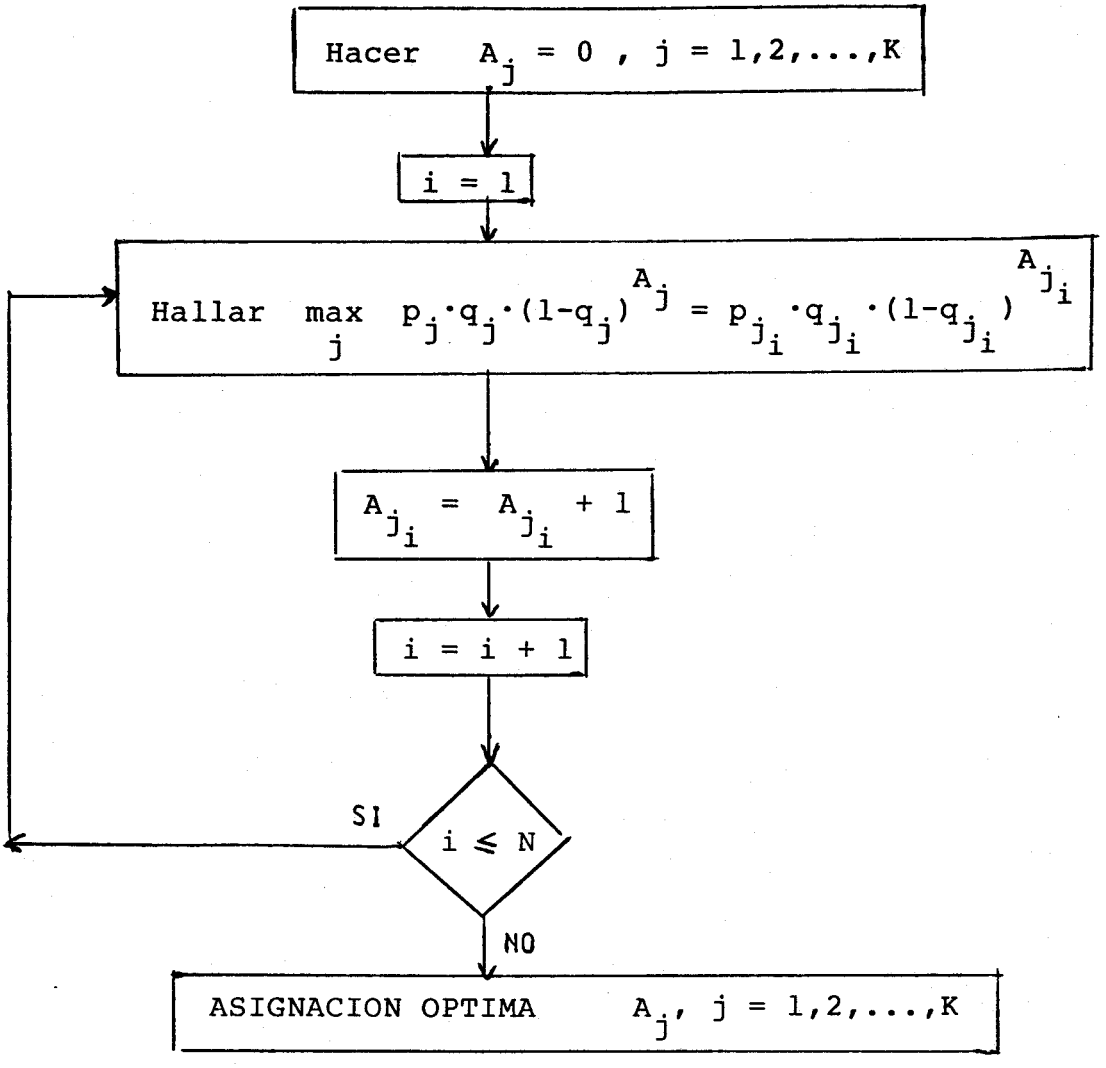
$$r \leq p_{j_i} \cdot \underline{P}(j_i, A(j_i, i)) = p_{j_i} \cdot \underline{P}(j_i, k) = p_{j_0} \cdot \underline{P}(j_0, k)$$

c.q.d.

Por tanto, dado N natural, la asignación óptima se obtiene mediante el siguiente algoritmo :



En nuestra situación particular quedaría,



II.5 APLICACION

Deseamos determinar el mínimo global entero de la función f definida sobre la malla entera constituida por los puntos de coordenadas enteras del interior de un rectángulo n-dimensional S.

Sea T un algoritmo que aplicado a la región $B \subset \mathbb{R}^n$ contenida en S, y generando m puntos sobre dicha región nos proporciona una estimación $T(B,m) \in B$ del mínimo global entero. Si el número total de puntos a generar por el algoritmo lo limitamos a que no exceda de un valor $N \in \mathbb{N}$ fijado por nosotros, caben dos estrategias a seguir:

1.- Obtener $T(S,N) \in S$

2.- Efectuar una partición de S constituida por K subrectángulos $S_1, S_2, \ldots, S_K$ de S ; descomponer N en suma de K sumandos $(N=N_1+\ldots,+N_K)$ y obtener $T(S_i,N_i) \in S$, $i=1,\ldots,K$.

Como estimación del mínimo tomaríamos la de menor valor funcional.

Si la partición efectuada sobre S la hacemos de tal forma que aquel subrectángulo S_i que contenga al mínimo sea precisamente al que le corresponda un N_i mayor y un menor volumen, entonces sería preferible la segunda estrategia.

En base a esto proponemos el siguiente algoritmo:

ALGORITMO COMBINADO

Paso 0 : Tomar $\gamma \in (0,1)$.

Paso 1 : Con $N_0 = N \cdot \gamma$ puntos, obtener información sobre la partición a efectuar sobre S y determinar las probabilidades p_i .

Paso 2 : Obtener la asignación $(N-N_0)$-óptima $A^{**}=(A_1^{**},\ldots,A_K^{**})$

Paso 3 : Obtener $T(S_i,A_i^{**}) \in S_i$, $i=1,2,\ldots,K$

Paso 4: Dar como estimación del mínimo global el de menor valor funcional.

Quedan por aclarar dos cuestiones para aplicar el algoritmo :

a) ¿Cómo llevar a cabo el paso 1?

b) ¿Es preferible la estimación T(S,N) a la proporcionada por el algoritmo combinado?. Dar un criterio de decisión.

Los subrectángulos los formaremos de distintos volúmenes de forma que el menor valor sea el candidato más probable a contener al mínimo global. Podemos proceder así :

Con N_0 puntos aplicamos el algoritmo entero de Price , (ver apéndice I), con lo cual obtendremos $N_0' \leq N_0$ puntos que tienden a agruparse en torno de los puntos mínimos glo-

bales. Mediante estos puntos efectuaremos la partición y estimaremos las probabilidaes p_i.

Dividimos el rectángulo S en dos subrectángulos S_1' y S_2' por el lado de mayor longitud, y calculamos la proporción de puntos que contienen de entre los N_0'. El de menor proporción será S_1 y el otro lo vovemos a dividir en dos. Este proceso lo realizamos mientras la proporción de puntos en cada mitad sea significativamente distinta o hasta haber determinado un número prefijado de subrectángulos. Obtenemos así una partición $S_1,\ldots,S_K$ de S de forma que :

$$\text{card}(S_i) = \text{card}(S)/2^i \quad , \; i=1,2,\ldots,K-1$$

$$\text{card}(S_K) = \text{card}(S)/2^{K-1}$$

Pasemos a calcular las probabilidades p_i :

Tras la partición efectuada parece lógico suponer que estas probabilidades a priori sean inversamente proporcionales al volumen del subrectángulo y directamente proporcionales a la proporción de puntos que contienen de entre los N_0'. Así pues tendremos que :

$$p_i = \alpha \cdot \frac{n_i}{\text{card}(S_i)} = \frac{\alpha \cdot n_i}{V_i}$$

Como $\sum p_i = 1$, deducimos que :

$$p_i = \frac{1}{\sum_j \frac{n_j}{V_j}} \cdot \frac{n_i}{V_i} \quad , \; i=1,2,\ldots,K$$

donde n_i es el número de puntos entre los N_0' que pertenecen a S_i.

Una vez calculados estos valores podremos determinar la asignación $(N - N_0)$-óptima, que es la que maximiza la probabilidad de detectar al mínimo, supuesto que T consista en la extracción uniforme de puntos.

Un criterio para elegir entre la estimación T(S,N) y la proporcionada por el algoritmo combinado sería calcular las probabilidades de detección para cada situación,

$$P_1 = 1 - (1 - 1/ \text{card}(S))^N \quad , \text{ para T.}$$

$$P_2 = \sum_j p_j \cdot (1 - (1 - 1/ \text{card}(S_j))^{A^{**}_j}) \quad \text{para el algoritmo combinado.}$$

y elegir la correspondiente al mayor valor.

Cuando T sea un algoritmo más elaborado que el considerado, todo lo anterior seguirá cumpliéndose siempre que las probabilidades de detección estén en la misma proporción que en las de aquél. Esto no lo sabremos en la práctica pero el algoritmo combinado ofrece excelentes resultados tal como se comprobará en el capítulo IV.

CAPITULO III : METODO DE LAS FUNCIONES PENALIZADORAS

III.1 INTRODUCCION

En capítulos anteriores hemos expuesto algunos algoritmos de tipo estocásticos para resolver el problema de la optimización global entera. En este capítulo no vamos a considerar un nuevo algoritmo, en el sentido estricto de la palabra, sino que abordaremos el problema de forma indirecta, transformando el problema en variables enteras en otro en variables reales, de forma que sus soluciones estén relacionadas de algún modo.

Para ello partiremos de la disponibilidad de un algoritmo que resuelva el problema continuo de optimización global, por ejemplo, el dado por Theodoor Timmer (20) basado en técnicas de análisis cluster.

Queremos resolver el siguiente problema de optimización entera :

$$P_e : \text{minimizar} \quad f(x)$$
$$x \in Z^n$$

donde f es una función real n-dimensional (que cumple las condiciones necesarias para poder aplicarle el algoritmo de minimización del cual partimos), y Z^n representa el conjunto de puntos de $\mathbb{R}^n$ con coordenadas enteras.

Suponemos que el problema posee solución finita, o sea,

$$\exists f^* \in \mathbb{R} \ , \exists n^* \in Z^n \quad / \quad f^* = f(n^*) = \underset{x \in Z^n}{\text{minimo}} \ f(x)$$

Asociado al problema P_e tenemos el siguiente problema continuo :

$$P : \text{minimizar} \quad f(x)$$
$$x \in \mathbb{R}^n$$

que se obtiene eliminando la condición de que la variable tome tan solo valores enteros.

Para funciones generales no existe ningún tipo de relación entre las soluciones de los problemas P_e y P.

Como sabemos el problema P_e es de gran complejidad, y una de las principales causas radica en que al estar evaluada la función en la malla entera, puede existir gran diferencia entre los valores funcionales de puntos enteros contiguos, siendo pues muy dificil de establecer la mejor dirección de búsqueda.

Parece intuitivo que un algoritmo eficiente para P_e debería basarse también en el comportamiento de la función sobre los puntos no enteros, es decir, el problema P_e debería resolverse via el problema P.

Nuestra idea para conseguir esto es pasar del problema P_e a otro continuo, de modo que los mínimos globales respectivos estén relacionados de algún modo : Sería ideal encontrar una función $\emptyset$ asociada a f de forma que, no diferiendo mucho de ella, gozáse de la propiedad de que su mínimo global continuo se alcanzase en n*.

Una manera de proceder sería penalizando los puntos no enteros, pero la penalización debe ser tal que ésta aumente sobre los puntos no enteros mientras mayor sea el valor funcional sobre ellos. Este es el problema que trataremos en el siguiente apartado.

III.2 MODELO LINEAL DE PENALIZACION

El concepto de penalización que manejaremos queda expresado en la siguiente definición :

Definición III.2.1

Una función real n-dimensional se dirá que es una función penalizadora si cumple las tres propiedades siguientes :

1.- A es continua
2.- $A(x) = 0 \quad , \quad \forall x \in Z^n$
3.- $A(x) > 0 \quad , \quad \forall x \notin Z^n$

Posteriormente consideraremos distintas funciones convenientes, pero de momento la supondremos dada.

Dado $r > 0$ consideremos la siguiente función :

$$\phi_r(x) = f(x) + r \cdot A(x) \quad , \quad \forall x$$

Nuestro objetivo será determinar el valor del parámetro r para que dicha función nos sea de utilidad. Para ello vamos a estudiar sus propiedades.

Teorema III.2.1

Dado $r > 0$ denotemos por x_r^* a un punto mínimo global continuo de ϕ_r. Entonces, si $x_r^* \in Z^n$ se tiene que x_r^* es un punto mínimo global entero de f.

-D-

Recordamos que en nuestro trabajo partimos de la existencia de los mínimos globales, continuo y entero, de la función f. Es fácil ver que entonces también existen los de ϕ_r.

Como $\phi_r(x_r^*) \leq \phi_r(x) \quad \forall x \in \mathbb{R}^n$, tendremos que,

$$f(x_r^*) + r \cdot A(x_r^*) \leq f(x) + r \cdot A(x) \ , \quad \forall x \in \mathbb{R}^n$$

En particular,

$$f(x_r^*) \leq f(x) + r \cdot A(x) \qquad \forall x \in Z^n$$

Es decir,

$$f(x_r^*) \leq f(x) \qquad \forall x \in Z^n$$

c.q.d.

Tenemos pues que si ϕ_r posee un mínimo global continuo en un punto con coordenadas enteras, entonces dicho mínimo es la solución del problema P_e.

Consideraremos ahora la función ϕ_r para valores crecientes del parámetro y veamos qué ocurre con sus mínimos globales.

Teorema III.2.2

Sea $\{r_m\}$ una sucesión creciente de números reales positivos, entonces se tiene que :

a) $\left\{ \phi_{r_m}(x_{r_m}^*) \right\}_m$ es monótona creciente.

b) $\left\{ A(x_{r_m}^*) \right\}_m$ es monótona decreciente.

c) $\left\{ f(x_{r_m}^*) \right\}_m$ es monótona creciente.

-D-

a) Por definición se tiene que,

$$\phi_{r_m}(x^*_{r_m}) \leq \phi_{r_m}(x^*_{r_{m+1}})$$

y , por tanto,

$$\phi_{r_m}(x^*_{r_m}) \leq f(x^*_{r_{m+1}})+r_m \cdot A(x^*_{r_m}) \leq f(x^*_{r_{m+1}})+r_{m+1} \cdot A(x^*_{r_{m+1}}) =$$

$$= \phi_{r_{m+1}}(x^*_{r_{m+1}})$$

b) Como,

$$\phi_{r_m}(x^*_{r_m}) \leq \phi_{r_m}(x^*_{r_{m+1}}) \qquad y \qquad \phi_{r_{m+1}}(x^*_{r_{m+1}}) \leq \phi_{r_{m+1}}(x^*_{r_m})$$

tendremos que,

$$f(x^*_{r_m})+r_m \cdot A(x^*_{r_m}) \leq f(x^*_{r_{m+1}})+r_m \cdot A(x^*_{r_{m+1}}) \quad , \text{ y además,}$$

$$f(x^*_{r_{m+1}})+r_{m+1} \cdot A(x^*_{r_{m+1}}) \leq f(x^*_{r_m})+r_{m+1} \cdot A(x^*_{r_m})$$

y, sumándolas obtenemos :

$$r_m \cdot A(x^*_{r_m}) + r_{m+1} \cdot A(x^*_{r_{m+1}}) \leq r_m \cdot A(x^*_{r_{m+1}})+r_{m+1} \cdot A(x^*_{r_m}) \quad ; \text{ o sea,}$$

$$(r_m - r_{m+1}) \cdot A(x^*_{r_m}) \leq (r_m - r_{m+1}) \cdot A(x^*_{r_{m+1}}) \quad ; \text{ y como } r_m \leq r_{m+1},$$

tendremos que,

$$A(x^*_{r_m}) \geq A(x^*_{r_{m+1}})$$

c) $\phi_{r_m}(x^*_{r_m}) \leq \phi_{r_m}(x^*_{r_{m+1}})$, y por tanto,

$f(x^*_{r_m}) - f(x^*_{r_{m+1}}) \leq r_m \cdot (A(x^*_{r_{m+1}}) - A(x^*_{r_m})) \leq 0$,debido a b).

luego,

$$f(x^*_{r_m}) \leq f(x^*_{r_{m+1}})$$

c.q.d.

A partir de ahora necesitaremos la hipótesis de que <u>la función objetivo crece indefinidamente en el exterior de una cierta, pero arbitraria, región</u>.

Esta hipótesis no es restrictiva, pues considerariamos una región lo suficientemente extensa para que en su interior se hallen los puntos mínimos globales; y bastaría con reedefinirla en el exterior de dicha región. Damos por tanto la siguiente definición :

<u>Definición III.2.2</u>

Se dirá que la función f se comporta como una función convexa en el exterior de la región S (que contiene a los puntos mínimos globales en su interior), si verifica que :

$$\lim_{\substack{x \longrightarrow +\infty \\ x \in \mathbb{R}^n}} f(x) = +\infty$$

Se tiene el siguiente importante resultado :

Teorema III.2.3

Supongamos que la función f es continua y que se comporta como una función convexa en el exterior de la región S.

Sea $\{r_m\}$ una sucesión monótona creciente y divergente de números reales positivos. Entonces la sucesión de mínimos globales $\{x^*_{r_m}\}$ posee una subsucesión convergente hacia un vector $\bar{x}$ de Z^n que es un punto mínimo global entero de f.

-D-

Como $\phi_{r_m}(x^*_{r_m}) \leq \phi_{r_m}(x)$, $\forall x \in \mathbb{R}^n$, $\forall m \in \mathbb{N}$, se tiene que,

$$f(x^*_{r_m})+r_m \cdot A(x^*_{r_m}) \leq f(x)+r_m \cdot A(x) \ , \ \forall x, \forall m \qquad (1)$$

Pero,

$$f(x^*_{r_m}) \leq f(x^*_{r_m})+r_m \cdot A(x^*_{r_m}) \leq f(x)+r_m \cdot A(x), \ \forall x, \forall m$$

luego,

$$f(x^*_{r_m}) \leq f(x) \ , \ \forall x \in Z^n, \forall m \in \mathbb{N} \qquad (2)$$

La sucesión $\{x^*_{r_m}\}$ debe estar acotada , pues de no ser así se tendría por hipótesis que, $\lim_{m \to \infty} f(x^*_{r_m}) = +\infty$

y esto contradice a lo obtenido en (2).

Por tanto, existirá una subsucesión convergente, que seguiremos denotándola por $\{x^*_{r_m}\}$ para simplificar la notación. O sea,

$$\exists \bar{x} \in \mathbb{R}^n \quad / \quad \lim_{m \to \infty} x^*_{r_m} = \bar{x}$$

Veamos que $\bar{x} \in Z^n$:

Necesariamente se ha de verificar que $\lim_{m \to \infty} A(x^*_{r_m}) = 0$

pues en caso contario $\exists\, m_0 \in \mathbb{N} \;/\; A(x^*_{r_m}) > \varepsilon > 0 \quad \forall m \geq m_0$

para un cierto ε .

(téngase en cuenta el apartado b) del terorema anterior)

Por tanto,

$$\lim_{m \to \infty} r_m \cdot A(x^*_{r_m}) = +\infty$$

y esto contradice al apartado c) del teorema anterior.

Por continuidad de A se tendrá que,

$$\lim A(x^*_{r_m}) = A(\bar{x}) = 0$$

y por tanto,

$$\bar{x} \in Z^n$$

Finalmente, como

$f(x^*_{r_m}) \leq f(x)$, $\forall x \in Z^n$, tendremos que,

$\lim f(x^*_{r_m}) \leq f(x)$, $\forall x \in Z^n$, y por continuidad de f,

$$f(\bar{x}) \leq f(x) \quad , \quad \forall x \in Z^n.$$

c.q.d.

En la demostración del teorema se ha probado que la sucesión de números reales $\{f(x^*_{r_m})\}$ posee una subsucesión convergente a f(n*), siendo n* un punto mínimo global de f. Teniendo en cuenta además el aprtado c) del teorema III.2.2 se obtiene el siguiente resultado :

Teorema III.2.4

Dada una sucesión $\{r_m\}$ de números reales no negativos, monótona creciente y divergente, considérese la sucesión de mínimos globales $\{x^*_{r_m}\}$ de las funciones ϕ_{r_m}.

Entonces se tiene que,

$$\lim_{m \to \infty} f(x^*_{r_m}) = f(n^*)$$

donde f es una función real n-dimensional continua, que se comporta como una función convexa en el exterior de una región S que contiene en su interior al mínimo global entero n^*.

-D-

Vimos que la sucesión $\{f(x^*_{r_m})\}$ poseía una subsucesión convergente hacia $f(n^*)$. Como sabemos que la propia sucesión es monótona creciente, se tiene que ella también converge al mismo límite.

c.q.d.

III.3 ALGORITMO DE LAS FUNCIONES PENALIZADORAS

Como consecuencia del apartado anterior, proponemos el siguiente algoritmo para resolver el problema P_e :

PASO 0 : Tomar una sucesión $\{r_m\}$ monótona divergente de números reales positivos.

PASO 1 : Hacer m=1

PASO 2 : Resolver el problema

$$\min_{x \in \mathbb{R}^n} \; (f(x) + r_m \cdot A(x))$$

y sea $x^*_{r_m}$ un punto mínimo global.

PASO 3 : Si $x^*_{r_m} \in Z^n$, entonces PARAR : $x^*_{r_m}$ es la solución del problema P_e.

En otro caso, hacer m = m + 1 , y volver al paso 2.

En cualquier iteración del algoritmo podemos disponer de una acotación del error que se comete si no continuamos : como estimación del mínimo global continuo se tomará

$$n^*_m = \text{INT} (x^*_{r_m})$$

formado por el vector cuyas coordenadas son las partes enteras de $x^*_{r_m}$.

Sabemos que se verificará la desigualdad,

$$f(x^*_{r_m}) \leq f(n^*) \leq f(n^*_m)$$

y por tanto, el error relativo que se cometerá al tomar a $f(n^*_m)$ como estimación de $f(n^*)$, vendrá dado por,

$$\frac{f(n^*_m) - f(x^*_{r_m})}{f(x^*_{r_m})} \cdot 100$$

Si el error, en tanto por ciento, lo acotamos por $E \in (0,100]$ el algoritmo quedaría :

Algoritmo de las funciones penalizadoras

PASO 0 : Tomar $r > 0$, $\Delta r > 0$.

PASO 1 : Determinar x^*_r , tal que, $\phi_r(x^*_r) = \min_{x \in \mathbb{R}^n} \phi_r(x)$

PASO 2 : Si $x^*_x \in Z^n$, entonces PARAR : x^*_r es el óptimo.

En otro caso, determinar:

$$n^*_r = \text{INT}\ (\ x^*_r\ ) \quad ,y \quad f(n^*_r)$$

PASO 3 : Si se verifica que ,

$$\frac{f(n^*_r) - f(x^*_r)}{f(x^*_r)} \cdot 100 < E$$

entonces PARAR: $f(n^*) \in (\ f(x^*_r), f(n^*_r))$.

En otro caso, hacer $r = r + \Delta r$ y volver al paso 1.

En cada iteración del algoritmo debemos realizar el paso 2, que es el que presenta gran dificultad, pues se debe resolver un problema de óptimización global continuo, aunque por hipótesis partimos que sabiamos resolverlo. Interesa pues no realizar muchas iteraciones, y por tanto se debe partir de un valor de r convenientemente grande. Pero, por otro lado, mientras mayor sea el parámetro , mayor será la dificultad al resolver el problema de optimización global continuo, pues la función $\emptyset_r$ va poseyendo más mínimos locales en los puntos de coordenadas enteras.

En consecuencia r deberá inicializarse en un valor moderado, en la confianza de que en pocas iteraciones la estimación obtenida esté dentro de los límites impuestos.

Posteriormente seguiremos con este problema, pero previamente vamos a considerar posibles funciones penalizadoras.

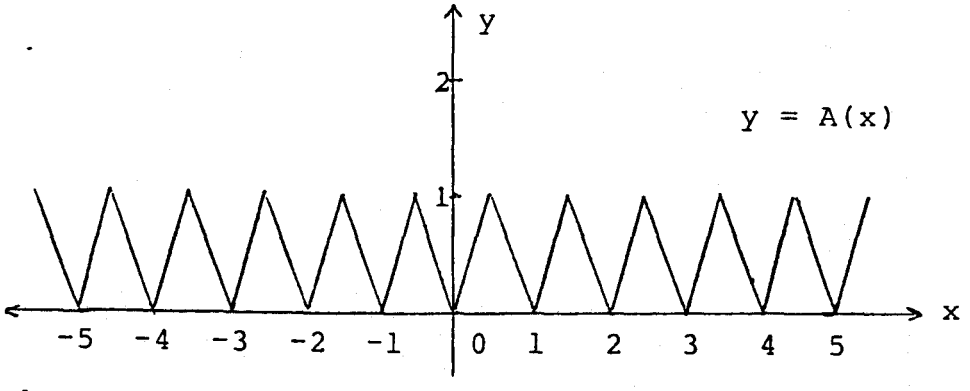
III.4 FUNCIONES PENALIZADORAS

Vamos a considerar el caso unidimensional (n=1). El valor A(x) debe ser mayor cuanto más apartado esté x de un entero; una posible función que cumpla este requisito podría ser :

$$\text{Si } x \geq 0 \text{ , } A(x) = \begin{cases} 2\cdot(x-INT(x)) & \text{si } INT(x) < x < INT(x)+\frac{1}{2} \\ 2\cdot(1-x+INT(x)) & \text{si } INT(x)+\frac{1}{2} < x < INT(x)+1 \end{cases}$$

Si $x < 0$, tomar $A(x) = A(-x)$.

Cuya representación gráfica es :



Evidentemente esta función es penalizadora, pues es continua, positiva, y se anula sobre los valores enteros. Además penaliza a los puntos proporcionalmente a la distancia que los separa de su parte entera. Tiene el inconveniente de que no es derivable en los puntos enteros, y por tanto la correspondiente función ϕ_r tampoco lo será.

Muchos algoritmos de búsqueda continua se basan en la diferenciabilidad de la función objetivo, e incluso en derivadas de orden superior, y por tanto no serán aplicables a dicha función. Vemos pues la nacesidad de encontrar otra función que cumpla este requisito.

Usando una función periódica, como por ejemplo la función seno, es fácil obtener una función penelizadora, como por ejemplo :

$$\underline{A}(x) = (1 + \text{sen}(2 \cdot \pi \cdot x - \pi/2))/2, \quad \forall x \in \mathbb{R}$$

Es trivial comprobar que esta función es penalizadora, está normalizada, es decir, toma valores en el intervalo unitario, y además es indefinidamente diferenciable. Por tanto, la función ϕ_r será tantas veces diferenciable como lo sea la función objetivo. Por todo lo cual ésta será la función penalizadora que consideraremos en la práctica.

Pasemos a continuación al caso n-dimensional :
Dado $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, consideremos las funciones vectoriales siguientes :

$$A_i(x) = A(x_i) \quad , i=1,2,\ldots,n$$

donde $A(\cdot)$ es una función penalizadora unidimensional. Nosotros tomaremos en la práctica $A = \underline{A}$.

Entonces, la función :

$$A^*(x) = \sum A_i(x)$$

es penalizadora para el caso vectorial, tomando valores en el intervalo [0,n] . Si queremos normalizarla, tomaríamos :

$$A^{**}(x) = A^*(x)/n$$

como función penalizadora.

III.5 CONSIDERACIONES SOBRE EL VALOR DEL PARAMETRO r

Sería deseable determinar, si existe, un valor de r tal que el mínimo global de $\emptyset_r$ sea entero, y por tanto dicho mínimo sería el mínimo global entero de f.

Habría pues que determinar $r>0$ tal que,

$$\emptyset_r(n^*) \leqslant \emptyset_r(x) \quad , \quad \forall x \in R^n$$

o sea,

$$f(n^*) \leqslant f(x) + r \cdot A(x) \quad , \forall x \in \mathbb{R}^n$$

Observése que dicha desigualdad se verifica para todo valor de r positivo siempre que,

$$x \in Z^n \qquad \text{o bien,} \quad f(n^*) \leqslant f(x) \quad \forall x \in \mathbb{R}^n/Z^n$$

Por tanto, el problema se reduce a encontrar un valor de r tal que :

$$f(n^*) \leqslant f(x) + r \cdot A(x) \quad , \forall x \in B$$

siendo $B = \{ x \in R^n/Z^n \text{ tal que } f(x) < f(n^*) \}$

Exceptuando el caso trivial de que n* sea mínimo global continuo de f, está claro que B no es vacio, debido a la continuidad de la función.

Tenemos pues la siguiente condición para que x_r^* sea un vector entero :

$$r \geqslant \underset{x \in B}{\text{supremo}} \frac{f(n^*) - f(x)}{A(x)}$$

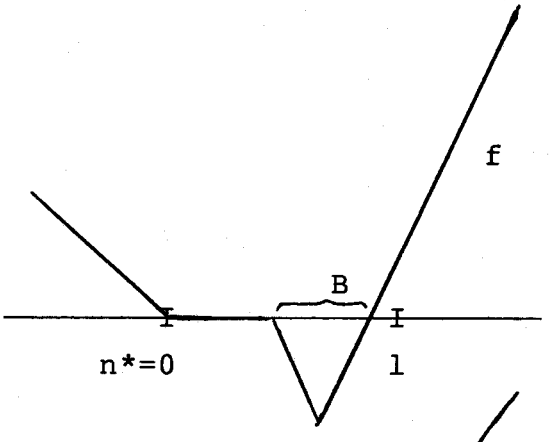
Si se verifica la siguiente condición :

$$\exists\, \varepsilon > 0 \quad \text{tal que,} \quad A(x) \geq \varepsilon \;, \quad \forall x \in B$$

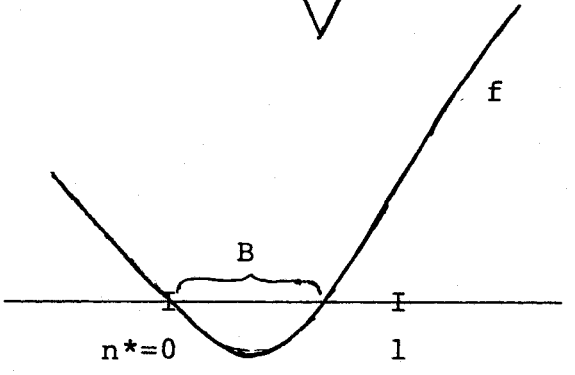
dicho supremo sería finito, y existiría el valor de r que buscamos. Pero si eso no ocurre; o sea, que el conjunto B contiene puntos tan cercanos a un vector de coordenadas enteras como queramos, entonces el supremo podría valer infinito y el valor de r no existir; que es lo que ocurre en la mayoría de las situaciones prácticas :

Ejemplo : (n=1)

.- Caso favorable ($r < \infty$)



.- Caso desfavorable ($r = \infty$)



Por tanto, al aplicar el algoritmo de las funciones penalizadoras, casi nunca lo detendremos en el paso 2 ($x_r^* \epsilon Z^n$), sino más bien en el paso 3.

Téngase en cuenta que realmente no necesitamos que x_r^* esté en Z^n, sino que esté lo suficientemente cerca de n*

como para que $n^* = INT(x^*_r)$. Teniendo esta idea en mente, vamos a dar un paso más.

Hemos partido al comienzo del capítulo de la hipótesis de que disponemos de un algoritmo o estrategia que nos permite localizar al mínimo global continuo de una función n-dimensional. Supongamos que dicho algoritmo también nos determina al mínimo si evaluamos la función en un conjunto discreto "suficientemente denso". Nótese que al implementar un algoritmo en el ordenador hacemos una discretización de $\mathbb{R}^n$, pues sólo se consideran números reales decimales exactos.

Así por ejemplo, para fijar ideas, consideremos una discretización de $\mathbb{R}^n$ en milésimas, indicando con ello que tomaremos el conjunto $\mathbb{R}^n_{1000}$ de $\mathbb{R}^n$ formado por aquellos vectores tales que sus componentes son números reales con una representación decimal de a lo sumo tres decimales.

Suponemos pues que sabemos resolver el problema :

$$\text{minimizar} \quad f(x)$$
$$x \in \mathbb{R}^n_{1000}$$

Con esto lo que hacemos es pasar de la malla entera a otra más densa, de forma que sobre ésta última sabemos minimizar la función objetivo (usando si es necesario una versión adaptada del algoritmo continuo) y a partir de ello obtener el mínimo global entero.

Es evidente que al reemplazar $\mathbb{R}^n$ por $\mathbb{R}^n_{1000}$ el algoritmo de las funciones penalizadoras no sufre alteración pues los teoremas en los que se basa siguen siendo válidos en la nueva situación.

En estas condiciones si existe un valor de r con la condición requerida :

Teorema III.5.1

Consideremos f evaluada tan solo sobre R^n_{1000}. Entonces existe un valor r_0 positivo tal que el mínimo global entero de f es mínimo global de ϕ_{r_0}.

-D-

Al igual que vimos anteriormente basta con determinar un valor de r tal que,

$$f(n^*) - f(x) \leq r \cdot A(x) \ , \ \forall x \in R^n_{1000}/Z^n \text{ y } f(x) < f(n^*)$$

Sea K_0 tal que $K_0 \leq f(x) \ \forall x \in R^n_{1000}$.

Por tanto si se cumple,

$$f(n^*) - K_0 \leq r \cdot A(x)$$

se cumplirá la condición anterior.

Como

$$A(\overbrace{0.001, 0.001, \ldots, 0.001}^{n}) = A(0.001^n) \leq A(x), \ \forall x \in R^n_{1000}$$

tendremos que la desigualdad requerida se verificará si tomamos un valor de r tal que,

$$f(n^*) - K_0 \leq r \cdot A(0.001^n)$$

Luego basta tomar,

$$r_0 \geq \frac{f(n^*) - K_0}{A(0.001^n)}$$

c.q.d.

Si denotamos por K_1 un valor de f sobre un vector entero, al tomar

$$r_0 = \frac{K_1 - K_0}{A(0.001^n)}$$

tendriamos garantizado que $x^*_{r_0}$ es un vector entero, con lo cual bastaría aplicar tan solo una iteración del algorimo de las funciones penalizadoras.

CAPITULO IV : RESULTADOS NUMERICOS

IV.1 RESULTADOS

En este capítulo presentamos los resultados numéricos obtenidos para los siguientes algoritmos considerados en la memoria :

A1 : ALGORITMO UNIFORME
A2 : ALGORITMO ENTERO DE PRICE
A3 : ALGORITMO RETICULAR
A4.K : ALGORITMO COMBINADO TOMANDO K SUBRECTANGULOS EN LA PARTICION.

Hemos tomado las siguientes funciones test :

FUNCION F1 :

$$f(x_1,x_2,x_3,x_4) = 100\cdot(x_2 - x_1^2)^2 + (1-x_1)^2 + 90\cdot(x_4 - x_3^2)^2 + \\ +10.1\cdot((x_2-1)^2 + (x_4-1)^2) + (1-x_3)^2 + \\ +19.8\cdot(x_2-x_1)\cdot(x_4-x_1)$$

sujeto a,

$$-10 \leq x_i \leq 10 \qquad i = 1,2,3,4$$

Mínimo global entero : (1,1,1,1) con un valor funcional 0

La región de muestreo contiene $21^4 = 194481$ puntos.

FUNCION F2 :

$$f(x_1,x_2,x_3) = 0.6\cdot x_1^2 - 59\cdot x_1 + 0.3\cdot x_2^2 - 23.5\cdot x_2 + 0.2\cdot x_3^2 - \\ - 28.5\cdot x_3$$

sujeto a,

$$0 \leq x_1 \leq 99$$
$$0 \leq x_2 \leq 76$$
$$0 \leq x_3 \leq 143$$

Mínimo global entero : (49,39,71) con un valor funcional igual a -2925.9

La región de muestreo contiene 1108800 puntos.

FUNCION F3 :

$$f(x_1,x_2,x_3,x_4) = x_1^2 + 10\cdot x_2^2 + 5\cdot(x_3-x_4)^2 + (x_2-2\cdot x_3)^4 + 10\cdot(x_1-x_4)^4$$

sujeto a , $-500 \leqslant x_i \leqslant 500 \quad i = 1,2,3,4$

Mínimo global entero : (0,0,0,0) con un valor funcional de 0.

La región de muestreo contiene $1001^4 \cong$ 1E+12 puntos.

FUNCION F4 :

$$f(x_1,x_2,x_3) = -16\cdot x_1 - 2\cdot x_2 - 9\cdot x_3$$

sujeto a, $0 \leqslant x_i \leqslant 99 \quad i = 1,2,3$

$$3\cdot x_1 + x_2 + 2\cdot x_3 \leqslant 100$$

Mínimo global entero : (33,1,0) y (32,0,2) con un valor funcional igual a -530.

La región de muestreo contiene 1000000 puntos.

FUNCION F5 :

$$f(x_1,x_2,x_3,x_4,x_5,x_6,x_7) = 13\cdot x_1 + 15\cdot x_2 + 14\cdot x_3 + 11\cdot x_4$$

sujeto a, $\qquad 0 \leq x_i \leq 20 \qquad i = 1,2,3,4,5,6,7$

$$4\cdot x_1 + 5\cdot x_2 + 3\cdot x_3 + 6\cdot x_4 - x_5 \geq 96$$

$$20\cdot x_1 + 21\cdot x_2 + 17\cdot x_3 + 12\cdot x_4 - x_6 \geq 200$$

$$11\cdot x_1 + 12\cdot x_2 + 12\cdot x_3 + 7\cdot x_4 - x_7 \geq 101$$

Mínimo global entero : (0,0,0,17,a,b,c)

con $\quad 0 \leq a \leq 6$

$\quad 0 \leq b \leq 4$

$\quad 0 \leq c \leq 18$

con un valor funcional igual a 187.

La región de muestreo contiene $21^7 = 1801088541$ puntos.

En todas estas funciones los valores de las variables están limitados a los valores enteros de un intervalo cerrado cuyo producto cartesiano, de todos ellos constituye, un rectángulo n-dimensional que denominaremos región de muestreo que es donde cada algoritmo genera sus puntos de búsqueda.

Obsérvese que las regiones de muestreo contienen un gran número de puntos.

A parte pueden existir restricciones en forma de inecuaciones.

Para estudiar el comportamiento de los algoritmos sobre las distintas funciones test hemos tomado como criterio de parada el detenerlos cuando se hayan generado un número dado M de puntos. En la evaluación de este número se consideran todos los puntos que cada algoritmo genera en cada iteración, sean factibles o no. Los valores de M los hemos limitado a 500,1000,1500 y 2000, independientemente del número de puntos de la región de muestreo de cada función, pues nos interesa estudiar la efectividad de cada algoritmo al generar un número no excesivo de puntos. Téngase en cuenta que para las funciones consideradas y para M = 2000 la fracción de muestreo máxima es del orden de 0.01.

Por tratarse de algoritmos de caracter estocástico, y poder comparar la efectividad de los mismos, para cada valor de M hemos realizado 10 repeticiones independientes de cada algoritmo, y como medida de efectividad hemos tomado la media aritmética de los valores funcionales de las estimaciones obtenidas, Asimismo hemos calculado la desviación típica de dichos valores para tener una medida de la dispersión de los mismos.

Los resultados obtenidos se presentan en las tablas que siguen. En ellas, para cada celda, el número superior es la media de los valores funcionales obtenidos, y el inferior la desviación típica. M es el número total de puntos generados, fijado por nosotros, y fr es la fracción de muestreo, o sea, el cociente entre M y el número de puntos que contiene la región de muestreo.

FUNCION F1

M / fr	A1	A2	A3	A4.2	A4.3
500	2433	36.48	44.76	8.88	23.98
2.6E-3	818.6	23.73	45.93	4.44	15.65
1000	2163.3	0.67	2.31	2.22	6.66
5.1E-3	221.18	1.49	12.30	4.46	5.44
1500	946.4	0	0	0	0
7.7E-3	528.9	0	0	0	0
2000	746.4	0	0	0	0
0.0103	283.9	0	0	0	0

FUNCION F2

M / fr	A1	A2	A3	A4.2	A4.3
500	-2573.9	-2916.16	-2925.75	-2925.84	-2925.70
4.5E-4	10.73	3.32	0.08	0.05	0.21
1000	-2780.08	-2923.86	-2925.9	-2925.9	-2925.9
9.0E-4	50.23	3.30	0	0	0
1500	-2751.32	-2925.9	-2925.9	-2925.9	-2925.9
1.3E-3	4.53	0	0	0	0
2000	-2763.52	-2925.9	-2925.9	-2925.9	-2925.9
1.8E-3	5.68	0	0	0	0

FUNCION F 3

M / fr	A1	A2	A3	A4.2	A4.3
500	821320.6	572324.3	264115.4	478701.2	510829.0
5E-10	272090.1	254200.1	205208.0	327483.9	298594.2
1000	581086.4	19314.4	5022.2	778.2	887.3
1E-09	86238.9	24437.4	9475.1	731.9	1347.0
1500	697058.1	2761.8	691.5	33.4	18.3
1.5E-09	149249.6	2429.8	1035.9	24.4	15.2
2000	654293.4	797.3	151.0	6.6	9.8
2E-09	76187.0	1279.8	334.9	5.4	6.9

FUNCION F_4

M / fr	A1	A2	A3	A4.2	A4.3
500	-410.4	..	-482.4	..	..
5E-04	20.4	..	14.9	..	..
1000	-438.6	-506.4	-512.0	-529.0	-526.1
1E-03	22.5	19.8	12.0	2.1	5.8
1500	-467.4	-523.2	-512.0	-528.1	-527.9
1.5E-03	11.3	5.4	18.1	3.4	4.6
2000	-449.1	-530	-530	-530	-530
2E-03	18.7	0	0	0	0

Nota: .. indica que el valor de M ha sido insuficiente.

FUNCION F 5

M / fr	A1	A2	A3	A4.2	A4.3
500	251.0	266.1	215.4	201.6	218.6
2.8E-07	6.4	32.3	16.8	15.5	21.7
1000	233.2	257.3	201.6	190.0	196.6
5.6E-07	1.4	4.5	14.0	4.1	7.5
1500	235.8	240.5	191.3	192.2	196.8
8.3E-07	4.7	27.5	2.0	4.7	7.1
2000	234.6	237.6	189.2	192.8	192.6
1.1E-06	5.2	17.4	4.4	4.2	4.5

IV.2 CONCLUSIONES

El algoritmo más simple de todos es el algoritmo uniforme, que no emplea información alguna en su búsqueda del mínimo, pero lo tomamos como base de comparación de los restantes algoritmos.

Como vimos en los capítulos anteriores, cada uno de aquellos depende de un conjunto de parámetros que hay que prefijar de antemano; además la efectividad de los mismos dependerá de la elección de dichos valores.

Así, para el algoritmo entero de Price dichos parámetros son el número total de puntos que debe contener la matriz cluster y la tasa de éxitos (27). Para el primero ha resultado efectivo un valor de 10 por variable, y el segundo lo hemos fijado en 0.8 para dar opción a una generación rápida de puntos secundarios.

En el algoritmo reticular los parámetros son RO,PAS,ET y FT: La semiamplitud del rectángulo de búsqueda RO la hemos fijado igual al semilado de menor amplitud de la región de muestreo para que de esta forma se puedan alcanzar la mayoria de los puntos de dicha región en las primeras iteraciones y para que el número de puntos no factibles del rectángulo de búsqueda no sea excesivo.

PAS indica el incremento/decremento con el que se modifica el rectángulo de búsqueda. Ha resultado conveniente el valor 1 ó 5.

ET y FT son respectivamente el número de éxitos y de fracasos que se deben alcanzar para modificar dicho rectángulo. Son los parámetros más importantes pues de ellos depende primordialmente el éxito de la búsqueda. El algoritmo nos ha resultado efectivo al darle a FT un valor doble al de ET, y respecto a éste el valor más apropiado ha sido 5, independientemente de la función que se considere. Incrementando este valor la búsqueda resultará más exhaustiva pero a costa de

necesitarse un mayor número de puntos para centrar al rectángulo de búsqueda sobre los puntos mínimos.

El algoritmo reticular ha resultado más efectivo que el algoritmo entero de Price en todos los casos.

En el algoritmo combinado presentado en el apartado II.5 son tres los parámetros : el número de subrectángulos K de la partición a efectuar, el algoritmo de búsqueda T que se aplicará en cada uno, y la proporción de puntos ν que se ha de tomar para calcular las probabilidades a priori.

Tras los resultados anteriores tomamos como T al algoritmo reticular. El algoritmo entero de Price lo hemos aplicado en el algoritmo combinado para aprovechar la información que nos ofrece la matriz cluster : la proporción de puntos de dicha matriz que haya en cada subrectángulo nos determinará la probabilidad a priori p_i.

El parámetro K es delicado. Su valor depende de los puntos que se hayan obtenido en la matriz cluster; si éstos no se han concentrado suficientemente entorno al mínimo, al tomar un valor de K excesivo se puede perder su localización. Hemos considerado los valores K=2 y K=3. La razón de no tomar valores superiores radica en cómo son las asignaciones óptimas de costo dado N : cuando el volumen de los subrectángulos es de gran magnitud y N toma un valor pequeño en relación a aquellos entonces la asignación óptima consiste en asignar todos los puntos al subrectángulo de menor volumen, y en caso de igualdad de éstos, al que posea mayor probabilidad a priori p_i. Es por esto por lo que no conviene tomar un valor superior a 3, a no ser que los puntos de la matriz cluster estén distribuidos alrededor del mínimo global.

Asimismo el parámetro ν es también importante. De él depende en gran medida la efectividad del algoritmo. ν debe ser tal que las probabilidades a priori que se obtengan aplicando $\nu \cdot M$ puntos con el algoritmo entero de Price sean suficien-

temente realistas para que la asignación óptima de los $(1-\gamma)\cdot M$ puntos restantes sea adecuada y suficiente para localizar al mínimo. Gracias a la rapidez con que el algoritmo entero de Price concentra los puntos de la matriz cluster no es necesario un valor de γ superior a 0.5, y así la fase de búsqueda del algoritmo es más exhaustiva. Nosotros hemos tomado los valores 0.25 ó 0.3.

Los resultados obtenidos con K=2 muestran que este algoritmo es superior en efectividad a los anteriores (de hecho ha tomado lo mejor de ellos).

Sin embargo con K=3 los resultados son del mismo orden, incluso en algunos casos es de menor efectividad. La razón de esto se debe a que al disminuir el tamaño de un subrectángulo a la mitad el algoritmo reticular no resulta más efectivo, siempre que este tamaño no sea suficientemente grande. Además el subrectángulo de menor volumen tiene de esta forma más posibilidades de no contener al mínimo.

Concluimos pues que el algoritmo que mejores resultados nos ha proporcionado ha sido el algoritmo combinado. Cabe destacar que evaluando 2000 puntos, y en algunas ocasiones con un número menor, siempre ha determinado el mínimo global entero de cada una de las funciones test en más de una ocasión de entre las diez repeticiones realizadas; obsérvese el pequeño valor de la desviación típica en cada caso y la proximidad de la media al mínimo. Además la efectividad del algoritmo puede ser incrementada via el algoritmo de búsqueda T que emplea. Nosotros tomamos el algoritmo reticular por ser el que mejor resultado nos ha dado.

Finalmente señalemos que los algoritmos que hemos considerado se basan en la distribución uniforme sobre la región factible. Como ésta es finita se implica que los puntos generados, por lo menos en las primeras iteraciones, se pueden considerar

como una muestra aleatoria simple.

Los procedimientos de muestreo para poblaciones finitas han sido estudiados con gran profusión, y cabe esperar que los algoritmos de búsqueda global que se basen en procedimientos más sofisticados que el aleatorio simple sean más eficientes. Habrá que buscar la forma de incorporar información previa sobre la función, localización de los mínimos, forma de la región factible, etc. de la cual dispongamos, mediante un procedimiento de muestreo apropiado. Esta es la linea de investigación que seguiremos en el futuro.

APENDICE I : ALGORITMO ENTERO DE PRICE

ALGORITMO ENTERO DE PRICE

El algoritmo de Price (26) es un procedimiento de caracter aleatorio para determinar el mínimo global de una función n-dimensional. Una versión adaptada para el problema de optimización global entera puede verse en (27), a la cual denominaremos algoritmo entero de Price.

La finalidad de éste es obtener un conjunto de puntos que se concentren entorno a algún punto mínimo global entero, y si es posible obtener entre ellos a uno de éstos. Para ello se parte de una matriz inicial de puntos enteros obtenidos de forma uniforme sobre la región factible (matriz cluster). A lo largo de las sucesivas iteraciones del algoritmo, estos puntos se van sustituyendo por otros de menor valor funcional en base a los denominados puntos primarios y secundarios,que constituyen la fase de búsqueda del algoritmo. Estos puntos dependen en cada iteración de la configuración actual de los puntos de la matriz cluster.

El algoritmo finaliza cuando la diferencia entre el mayor y el menor valor funcional de los puntos en la matriz es inferior a un valor prefijado, o bien, cuando se hayan generado un número dado de puntos.

Con este algoritmo se obtienen buenas estimaciones del mínimo global entero y además es fácil de implementar en un pequeño ordenador personal (27).

El motivo de considerar este algoritmo en la memoria es doble; por un lado ofrece unos buenos resultados que nos pueden servir de comparación para otros algoritmos, y por otro lado lo utilizamos como fuente de información sobre las probabilidades a priori de los subrectángulos que forman la partición de la región de muestreo via los puntos de la matriz cluster.

APENDICE II : PROGRAMAS

Los resultados presentados en las tablas anteriores los hemos obtenido mediante un ordenador personal PC-INVES 630 TURBO II. El lenguaje de programación utilizado ha sido el GWBASIC de Microsoft, soportado por el sistema operativo MS-DOS.

Presentamos a continuación el listado del programa que ejecuta el algoritmo combinado. Para cada nueva función que se considere han de modificarse las lineas 2060 y 2130.

Los restantes listados de los algoritmos que hemos utilizado no los hemos incluido pues básicamente corresponden a las distintas subrutinas del programa que tratamos.

```
10 REM ************************************************************************
20 REM *              ALGORITMO                    COMBINADO                    *
30 REM ************************************************************************
40 REM
50 REM                    **********************
60 REM                    * PROGRAMA PRINCIPAL *
70 REM                    **********************
80     RANDOMIZE TIMER
90     REM
100     REM           SE SOLICITAN  DATOS
110     REM
120     CLS:INPUT "Número de variables ";D
130     DIM R(D,2),X(D),R1(D,2),P1(D),Q1(D),SI(D+1,D),G1(D)
140     PRINT "Dar intervalos "
150     FOR I=1 TO D:INPUT R(I,1),R(I,2):NEXT
160     INPUT "Número total de puntos a generar ";N
170     INPUT "¿cuántos son para determinar las probabilidades a priori";M
180    N=N-M 'NUMERO DE PUNTOS PARA EL ALGORITMO DE BUSQUEDA
190    INPUT "¿Cuántos puntos en la matriz cluster";AP
200    INPUT "¿Hay restricciones, aparte de las del rectángulo (s/n)";W$
210    DIM A(AP,D+1),A1(AP,D+1),AA(D+1)
220    REM
230    REM Aplicaremos price en rectángulo inicial con un total de M puntos
240    REM para efectuar partición y calcular probabilidades a priori
250    REM Datos de entrada para la subrutina Price
260    REM
270    T1=M:N1=AP:FOR I=1 TO D:FOR J=1 TO 2:R1(I,J)=R(I,J):NEXT J:NEXT I
280    GOSUB 1090
290    FOR I=1 TO D+1:AA(I)=A1(I1,I):NEXT I
300    IF B1>M THEN CLS:PRINT CHR$(7):PRINT "La  región  factible  es dificil
       de muestrear. Con";M;"puntos no se ha podido generar matriz cluster":
       STOP
310    REM Se efectúa la partición
320    GOSUB 2150
330    REM
340    REM La partición de R(D,2) efectuada está en PART(K4,D,2)
350    REM
360    REM Cálculo de las probabidades a priori de los subrectángulos
370    DIM PR(K4),VO(K4)
380    FOR I=1 TO K4
390    : S=1
400    : FOR J=1 TO D:S=S*(PART(I,J,2)-PART(I,J,1)+1):NEXT J
410    : VO(I)=S
420    NEXT I
430    S#=0
440    :FOR I=1 TO K4:S#=S#+P4(I)/VO(I):NEXT I
450    FOR I=1 TO K4 :PR(I)=(1/S#)*(P4(I)/VO(I)):NEXT I
460    PRINT:PRINT "Las probabilidades a priori son : "
470    FOR I=1 TO K4
480    :PRINT "Subrectángulo ";I;" :";PR(I)
490    NEXT I
500    REM Determinación de la asignación N-óptima
510    DIM RI(K4),VE#(K4)
520    FOR I=1 TO N
530    : FOR J=1 TO K4
```

```
540   : VE#(J)=(PR(J)/VO(J))*(1-1/VO(J))^RI(J)
550   : NEXT J
560   : GOSUB 1000
570    : RI(II)=RI(II)+1
580   NEXT I
590   PRINT CHR$(7):PRINT "La asignación óptima es:"
600   FOR I=1 TO K4:PRINT "Celda";I;":",RI(I);" puntos":NEXT I
610   REM En el rectángulo PART(j,---) se deben muestrear RI(j) puntos
620   PRINT CHR$(7):PRINT "La probabilidad de detección es ",
630   FOR I=1 TO K4:H#=H#+PR(I)*(1-(1-1/VO(I))^RI(I)):NEXT
640   PRINT USING "#.#####";H#
650   REM En cada subrectángulo se aplica el algoritmo reticular
660   DIM R5(D,2),Y(K4,D+1),Y5(D),PS5(D)
670   FOR Z=1 TO K4
680   : REM Datos de entrada a la subrutina reticular
690   : IF RI(Z)=0 THEN Y(Z,D+1)=3.1415:GOTO 750
700   : FOR I=1 TO D:FOR J=1 TO 2:R5(I,J)=PART(Z,I,J):NEXT J:NEXT I
710   : NP5=RI(Z)
720   : GOSUB 2630
730   : FOR J=1 TO D:Y(Z,J)=Y5(J):NEXT J
740   : Y(Z,D+1)=FY5
750   NEXT Z
760   REM
770   REM Mostrar los resultados
780   REM
790   PRINT CHR$(7):CLS
800   PRINT "El resultado es :":FOR I=1 TO 20000:NEXT
810   FOR I=1 TO K4
820   : CLS:PRINT:PRINT " subrectángulo";I;":":PRINT:PRINT
830   : FOR K=1 TO D
840   : FOR J=1 TO 2
850   : PRINT PART(I,K,J),
860   : NEXT J
870   : PRINT
880   : NEXT K
890   : PRINT:PRINT "Se han generado",RI(I),"puntos"
900   : PRINT:PRINT:IF Y(I,D+1)=3.1415 THEN PRINT "NO SE HA MUESTREADO":
        GOTO 940
910   : PRINT "El mínimo es : "
920   : FOR K=1 TO D:PRINT Y(I,K),:NEXT K:PRINT
930   : PRINT "Valor funcional=",Y(I,D+1)
940   : PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
950   : PRINT CHR$(7):PRINT "Pulse PAUSA y después cualquier tecla"
960   : FOR W=1 TO 10000:NEXT W
970   NEXT I
980   END
990   REM
1000   REM Subrutina cálculo máximo de VE#(k4)
1010   II=1:ZA#=VE#(1)
1020   FOR C=2 TO K4
1030   : IF ZA#>=VE#(C) THEN 1050
1040   : II=C:ZA#=VE#(C)
1050   NEXT C
1060   RETURN
1070   REM   FINAL PROGRAMA PRINCIPAL
```

```
1080   REM
1090  REM ************************************************************************
1100  REM *            SUBRUTINA          PRICE                                  *
1110  REM ************************************************************************
1120  REM
1130  REM
1140  REM Los datos de entrada son :
1150  REM N1 = Número de puntos en matriz cluster
1160  REM T1 = Número total de puntos a generar por la subrutina
1170  REM R1 = Rectángulo de definición de las variables
1180  B1=0:K1=0:P1=0:S1=0:C1=0:O1=0:A1=1
1190  REM Creación de la matriz cluster
1200  FOR I=1 TO N1
1210  : FOR J=1 TO D
1220  : X(J)=INT(R1(J,1)+RND*(R1(J,2)-R1(J,1)+1))
1230  : NEXT J
1240  : B1=B1+1
1250  :IF W$="n" THEN I2=1 ELSE GOSUB 2000
1260  : IF I2=0 THEN IF B1>T1 THEN RETURN ELSE 1210
1270  : FOR J=1 TO D:A1(I,J)=X(J):NEXT J
1280  : GOSUB 2100
1290  : A1(I,D+1)=F1
1300  NEXT I
1310  REM Cálculo del menor y mayor elementos en la matriz
1320  ME1=A1(1,D+1):MA1=ME1:I1=1:J1=1
1330  FOR I=2 TO N1
1340  : IF A1(I,D+1)<ME1 THEN 1370
1350  : IF A1(I,D+1)<=MA1 THEN 1380
1360  : MA1=A1(I,D+1):J1=I:GOTO 1380
1370  : ME1=A1(I,D+1):I1=I
1380  NEXT I
1390  REM Simplex
1400  GOSUB 1960
1410  IF K1=1 THEN RETURN
1420  FOR I=1 TO D+1
1430  : R1=1+INT(N1*RND)
1440  : FOR J=1 TO D
1450  : SI(I,J)=A1(R1,J)
1460  : NEXT J
1470  NEXT I
1480  REM Centroide
1490  FOR I=1 TO D
1500  : G1(I)=0
1510  : FOR J=1 TO D
1520  : G1(I)=G1(I)+SI(J,I)
1530  : NEXT J
1540  : G1(I)=G1(I)/D
1550  NEXT I
1560  C1=C1+1
```

```
1570  GOSUB 1960
1580  IF K1=1 THEN RETURN
1590  REM Punto primario
1600  FOR I=1 TO D
1610  : P1(I)=INT(2*G1(I)-SI(D+1,I)+.5)
1620  : X(I)=P1(I)
1630  NEXT I
1640  P1=P1+1
1650  GOSUB 1910
1660  IF IN=0 THEN O1=O1+1:GOTO 1400
1670  IF W$="n" THEN I2=1 ELSE GOSUB 2000
1680  IF I2=0 THEN O1=O1+1:GOTO 1400
1690  GOSUB 2100
1700  FP=F1
1710  IF FP<A1(J1,D+1) THEN A1=A1+1:O1=O1+1:GOTO 1850
1720  O1=O1+1:TASA=A1/O1 'TASA DE EXITOS
1730  IF TASA>.8 THEN 1400
1740  REM Punto secundario
1750  S1=S1+1
1760  FOR I=1 TO D
1770  : Q1(I)=INT((G1(I)+SI(D+1,I))/2+.5)
1780  :X(I)=Q1(I)
1790  NEXT I
1800  IF W$="n" THEN I2=1 ELSE GOSUB 2000
1810  IF I2=0 THEN 1400
1820  GOSUB 2100
1830  FP=F1
1840  FOR I=1 TO D:P1(I)=Q1(I):NEXT I
1850  FOR I=1 TO D:A1(J1,I)=P1(I):NEXT I
1860  A1(J1,D+1)=FP
1870  GOSUB 1960
1880  IF K1=1 THEN RETURN
1890  GOTO 1320
1900  REM Se comprueba que el vextor X está en el rectángulo
1910  FOR I=1 TO D
1920  : IF X(I)<R1(I,1) OR X(I)>R1(I,2) THEN IN=0:GOTO 1950
1930  NEXT I
1940  IN=1
1950  RETURN
1960  REM Subrutina totalizadora de puntos generados
1970  IF B1+P1+S1+C1>T1 THEN K1=1
1980  RETURN
1990  REM FINAL DE LA SUBRUTINA PRICE
```

```
2000  REM ************************************************************************
2010  REM *         SUBRUTINA     DE       FACTIBILIDAD                           *
2020  REM ************************************************************************
2030  REM
2040  REM No se considera la definición del rectangulo
2050  REM Si el vector X es factible se devuelve I2=1, en otro caso I2=0
2060  IF 3*X(1)+X(2)+2*X(3)>100 THEN 2080
2070  I2=1:GOTO 2090
2080  I2=0
2090  RETURN
2100  REM ************************************************************************
2110  REM *         SUBRUTINA           FUNCIONAL                                  *
2120  REM ************************************************************************
2130  F1=-16*X(1)-2*X(2)-9*X(3)
2140  RETURN
2150  REM ************************************************************************
2160  REM *         SUBRUTINA     DE   PARTICION                                  *
2170  REM ************************************************************************
2180  REM
2190  DIM C41(AP,D),C42(AP,D),C43(AP,D),R41(D,2),R42(D,2),R43(D,2)
2200  PRINT CHR$(7)
2210  CLS:INPUT "Dar número se subrectángulos que formarán la partición";K4
2220  K4=K4-1:P4=AP
2230  DIM PART(K4+1,D,2),P4(K4+1)
2240  IF K4=1 THEN 2330
2250  FOR ETAPA=1 TO K4-1
2260  : FOR I=1 TO P4:FOR J=1 TO D:C43(I,J)=A1(I,J):NEXT J:NEXT I
2270  : FOR I=1 TO D:FOR J=1 TO 2:R43(I,J)=R(I,J):NEXT J:NEXT I
2280  : PP4=P4
2290  : GOSUB 2420
2300  : IF PUNTOS1<PUNTOS2 THEN P4=PUNTOS2:FOR I=1 TO P4:FOR J=1 TO D:  A1(I,J)=
C42(I,J):NEXT J:NEXT I:P4(ETAPA)=PUNTOS1:FOR I=1 TO D:   FOR J=1 TO 2:
        PART(ETAPA,I,J)=R41(I,J):R(I,J)=R42(I,J):NEXT J:NEXT I:GOTO 2320
2310  : P4=PUNTOS1:FOR I=1 TO P4:FOR J=1 TO D:A1(I,J)=C41(I,J):NEXT J:NEXT I:
        P4(ETAPA)=PUNTOS2:FOR I=1 TO D:FOR J=1 TO 2:PART(ETAPA,I,J)=R42(I,J):
        R(I,J)=R41(I,J):NEXT J:NEXT I
2320  NEXT ETAPA
2330  REM La última división se efectúa aparte
2340  FOR I=1 TO P4:FOR J=1 TO D:C43(I,J)=A1(I,J):NEXT J:NEXT I
2350  FOR I=1 TO D:FOR J=1 TO 2:R43(I,J)=R(I,J):NEXT J:NEXT I
2360  PP4=P4:GOSUB 2420
2370  P4(K4)=PUNTOS1:P4(K4+1)=PUNTOS2
2380  FOR I=1 TO D:FOR J=1 TO 2:PART(K4,I,J)=R41(I,J):PART(K4+1,I,J)=R42(I,J):
      NEXT J:NEXT I
2390  K4=K4+1
2400  PRINT CHR$(7):CLS:FOR I=1 TO K4:PRINT "Subrectángulo",I,"puntos",P4(I):
      NEXT I
```

```
2410  RETURN 'SE VUELVE AL PROGRAMA PRINCIPAL
2420  REM Cálculo del intervalo de mayor amplitud de R43
2430  AMPLITUD=R43(1,2)-R43(1,1):IN=1
2440  IF D=1 THEN 2500
2450  FOR I=1 TO D
2460  : AMPLITUD1=R43(I,2)-R43(I,1)
2470  : IF AMPLITUD<AMPLITUD1 THEN AMPLITUD=AMPLITUD1:IN=I
2480  NEXT I
2490  REM El rectángulo R43 actual se divide en dos por el lado de mayor
          amplitud
2500  FOR I=1 TO D
2510  : IF I<>IN THEN R41(I,1)=R43(I,1):R41(I,2)=R43(I,2):R42(I,1)=R43(I,1):
        R42(I,2)=R43(I,2):GOTO 2540
2520  : R41(I,1)=R43(I,1):R42(I,2)=R43(I,2)
2530  : R41(I,2)=R43(I,1)+AMPLITUD/2:R42(I,1)=R41(I,2)
2540  NEXT I
2550  REM Los puntos de C43 se van asignando a C41 ó C42 según corresponda
2560  PUNTOS1=0:PUNTOS2=0
2570  FOR I=1 TO P4
2580  : IF C43(I,IN)<=R41(IN,2)    THEN    PUNTOS1=PUNTOS1+1:  FOR J=1 TO D:
      C41(PUNTOS1,J)=C43(I,J):NEXT J:GOTO 2600
2590  : PUNTOS2=PUNTOS2+1:FOR J=1 TO D:C42(PUNTOS2,J)=C43(I,J):NEXT J
2600  NEXT I
2610  RETURN
2620  REM FINAL DE LA SUBRUTINA PARTICION
2630  REM ************************************************************************
2640  REM *        ALGORITMO                          RETICULAR                  *
2650  REM ************************************************************************
2660  REM
2670  REM       Datos de entrada :
2680  REM                    R5(D,2) = Rectángulo donde aplicar algoritmo
2690  REM                    RO5     = Semiamplitud=(lado mayor de R5)/4
2700  REM                    NP5     = Número total de puntos a generar
2710  REM Determinación del lado menor
2720  U=R5(1,2)-R5(1,1):LADO=U
2730  FOR I=2 TO D
2740  : U5=R5(I,2)-R5(I,1)
2750  : IF U5>U THEN 2770
2760  :U=U5:LADO=U
2770  NEXT I
2780  REM Semiamplitud automática
2790  RO5=INT(LADO/2)
2800  REM Paso automático
2810  IF RO5<=100 THEN PASO=1:GOTO 2840
2820  IF RO5>100 AND RO5<=500 THEN PASO=5:GOTO 2840
2830  IF RO5>500 AND RO5<=1000 THEN PASO=10 ELSE PASO=20
2840  ET5=5:FT5=10
2850  FOR I=1 TO D:X(I)=AA(I):NEXT
2860  GOSUB 3140
2870  IF IN5=0 THEN 2910
2880  FOR I=1 TO D:Y5(I)=AA(I):NEXT I
2890  FY5=AA(D+1):PRINT "EXITO=";FY5
2900  GOTO 2980
```

```
2910  FOR I=1 TO D:Y5(I)=INT(R5(I,1)+RND*(R5(I,2)-R5(I,1)+1)):X(I)=Y5(I)  :
       NEXT I
2920  GOSUB 3140
2930  IF IN5=0 THEN 2910
2940  GOSUB 2000
2950  IF I2=0 THEN 2910
2960  GOSUB 2100
2970  FY5=F1
2980  DEF FNF5(X)=INT(X-RO5+(1+2*RO5)*RND)
2990  WHILE (RO5<=1000000!) AND (B5<=NP5)
3000  : FOR I=1 TO D:PS5(I)=FNF5(Y5(I)):X(I)=PS5(I):NEXT I
3010  : GOSUB 3140
3020  : IF IN5=0 THEN 3110
3030  : GOSUB 2000
3040  : IF I2=0 THEN 3110
3050  : GOSUB 2100
3060  : FPS5=F1:IF FPS5<FY5 THEN PRINT FPS5,B5,"paso=";RO5
3070  : IF FPS5<FY5 THEN FOR I=1 TO D:Y5(I)=PS5(I):NEXT I:E5=E5+1:F5=0:
        FY5=FPS5  ELSE E5=0:F5=F5+1:GOTO 3090
3080  : IF E5>=ET5 THEN RO5=RO5+PASO:GOTO 3100
3090  : IF F5>=FT5 THEN RO5=RO5-PASO
3100  : IF RO5<=0 THEN RO5=3
3110  : B5=B5+1
3120  WEND
3130  RETURN
3140  REM Se comprueba que el vector X está en el rectángulo R5
3150  FORW=1 TO D
3160  : IF X(W)<R5(W,1) OR X(W)>R5(W,2) THEN 3190
3170  NEXT W
3180  IN5=1:GOTO 3200
3190  IN5=0
3200  RETURN
```

BIBLIOGRAFIA

(1) Bazara & Shetty (19). "Nonlinear Programming : Theory and Algorithms". John Wiley & Sons.

(2) Dixon,L.c.W.(1978). "Global optima without convexity". Technical Report. Numerical Optimization Centre, Hatfield Polytechnic. England.

(3) Evtushenko,Y.P. (1971). "Numerical Methods for finding global extrema of a nouniform mash". U.S.S.R. Computing Machines and Mathematical Physics 11,1390-1403.

(4) Shubert, B.O. (1972) "A sequential method seeking the global maximum of a function". Siam Journal on Numerical Analysis 9,379-388.

(5) Branin, F.H. (1972). "A methos for finding multiple estrema of a fuction of n variables". Numerical Methods of Nonlinear Optimization. Academis Press. London.

(6) Hardy, J.W. (1975). "An implemented extension of Branin s method". En Towards Global Optimization 2. North-Holland, Amsterdam.

(7) Goldstein, A.A. & Price, J.F. (1971). "On descent from local minima". Mathematics of Computations 25,569-574.

(8) Levy, A. & Gomez, S. (1980). "The tunneling algorithm for the global optimization problem of constrained function". Technical Report, Universidad Autónoma de Mexico.

(9) Solis, F.J. & Wets, R.J.E. (1981). "Minimizations by random search technics". Mathematics of Operations Research, 6,19-30.

(10) Rubinstein, R.Y. & Samorodnitsky, G. (1982). "Efficiency od the random search method". Math. and Comp. in Sim. 24.

(11) Wolfe, P. (1969). "Converge conditions for ascent methods". Siam Review 11,226-235.

(12) Dixon, L.c.W. (1980). "Nonlinear Optimization". Birkhaüser.Boston.

(13) Brooks, S.H. (1958). "A discussion of random methods for seeking maxima". Operations Research 6,244-251.

(14) Andersen, R.S. (1972). "Global Optimization in Optimization". University of Qeensland Press.

(15) Devroye, L. (1978). "Progressive global random search of continous fuctions". Mathematical Programming 15,330-342.

(16) Rubinstein, R.Y. (1981). "Simulation and the Monte Carlo method". John Wiley and sons,New York.

(17) Hartigan, J. (1975) "Clustering Algorithms". Wiley .

(18) Törn, A.A. (1976). "Cluster analysis using seed points and density determined hyperspheres with an application to global optimization". En Proceeding of the 3rd. International Conference on Pattern Recognition. Coronado. California.

(19) Törn, A.A. (1978). "A searh clustering approach to global optimization". En Towards Global Optimization. Dixon et al. North Holland. Amsterdam.

(20) Timmer, G.T. (1984). "Global Optimization : A Stochastic Approach". Centrum Voor Wiskunde en Informatica. Amsterdam. TESIS.

(21) Jiménez, M.T. (1984). "Metodos de Optimización Global". Memoria presentada para aspirar al grado de licenciado. Facultad de Matemáticas. Universidad de Sevilla.

(22) Taha, H.A. (1975). "Integer Programming : Theory, Applications and Computations". Academic Press.

(23) Conley, W. (1980). "Computer Optimization Techniques". A Petrocelli Book. New York. Princenton.

(24) Cooper, M.W. (1981). "A Survey of Methods for Pure Nonlinear Integer Programming". Management Sciences. Vol 27. No.3.

(25) Bazaraa, M.S. & Jarvis, J. (1978). "Linear Programming and Network Flows". Ed. John Wiley and Sons.

(26) Price, W.L. "A contolled Random Search Procedure for Global Optimization". (1978). In Toward Global Optimization I.

(27) Fernández García, F. & Ojo Mesa, J. & Pelegrin Pelegrin, B. (1985). "Un algoritmo de Optimización Global para Programación Entera". X Jornadas Hispano-Lusas de Matemáticas. Universidad de Murcia.

INDICE

UNIVERSIDAD DE SEVILLA

Reunido el Tribunal integrado por los abajo firmantes en el día de la fecha, para juzgar la Tesis Doctoral de D. Juan del Ojo Mesa titulada Métodos de multistart para la optimización global entera acordó otorgarle la calificación de APTO cum Laude

Sevilla, 22 de diciembre 1.988

El Vocal, El Vocal, El Vocal,

El Presidente El Secretario, El Doctorado,