UNIVERSIDAD CARLOS III DE MADRID

Departamento de Tecnologías de las Comunicaciones

TESIS DOCTORAL

"OPTIMIZACIÓN DE CUANTIFICADORES VECTORIALES BASADA EN ALGORITMOS GENÉTICOS Y TÉCNICAS HEURÍSTICAS"

Autor: Armando Malanda Trigueros
Director: Aníbal Ramón Figueiras Vidal

Leganés, 1999

Dedicada

a mis padres Angel y Adela,

a mis hermanos Angel, Fernando y Miguel

y a Elena.

AGRADECIMIENTOS

Quisiera recoger en estas líneas a todos aquéllos de quien he recibido ayuda, aliento o inspiración en la confección de esta Tesis. En realidad esta tarea se me presenta complicada, porque no me gustaría dejar a nadie olvidado y, sin embargo, de muchos soy deudor.

El primer agradecimiento es para mi Director, Aníbal Figueiras, a quien debo mucho más que la guía supervisión de mi Tesis y al que atribuyo el mérito de tenerme presente y confiar en mí, a pesar de que el tiempo, la distancia y el tortuoso camino recorrido no ayudaran demasiado.

En segundo lugar (advierto que no seguiré orden alguno, ni de importancia ni cronológico), citaré a mis primeros compañeros del Doctorado en Teleco de Madrid. Los Weruaga, Jesús Cid, Jantonio, Nacho, Panta, Javier García Frías, Ana, Fernando, Pedro y Luis Castedo (Nieves incluida), a los que más tarde se unieron Angel Navia, Carlos Bousoño y J. L. Sancho. Era el comienzo de mi Doctorado, allá a lo lejos en el tiempo; un tiempo en el que entre libros y estaciones de trabajo, había siempre espacio para bromas y tertulias; en el que los "papers" y los Cursos de Doctorado también dejaban tiempo para el partidillo de los jueves o las cañas de las ocho; un tiempo, en fin, de ilusión e incipiente amistad.

El turno ahora para el periplo londinense: gracias a Gerry y Annush por acogerme, a Liza, por su sin igual simpatía, a Owl por su admirable paciencia enseñándome los inalcanzables vericuetos del C, a Izzet, por su amistad, y a todo el CMSA, en general, por recibirme de una forma tan cálida. No olvido tampoco a la *Pequeña Eslovenia* (Yuri, Adrei, Issidor y Vani), ni, por supuesto, a Adam quien puso a mi disposición todo lo suyo, desde su habitación hasta sus calcetines, incluida buena parte de su escaso tiempo.

Volviendo a nuestra tierra y marchando atrás en el tiempo, aquí va mi sincera deuda de gratitud con Andrés, Germán, Jose y Lorenzo, compañeros de fatigas en Teleco, con quienes compartí noches de estudio y jolgorios, trabajos y huelgas, exámenes y vacaciones, risas y llantos en unos años maravillosos.

Pamplona es mi tierra adoptiva y el lugar donde he desarrollado la mayor parte de esta Tesis y mi labor profesional. Aunque en lo técnico he caminado casi siempre solo, no así en lo humano: David, Juan Ignacio, Carlos y un largo etcétera, en la Universidad, Esther , Susana ... fuera de ella. Y siempre, los alumnos. Aun tentado de citar el nombre de aquéllos de quienes guardo más estima, cariño o devoción, prefiero dejar a un lado

identidades y agradecer al alumno en su conjunto, motivación permanente del docente y causa inagotable de nuestro desquicio...

Mencionaba al comienzo mis deseos de agradecer a aquéllos que han sido causa de mi inspiración. Ello me conduce a Madrid, en busca de cuatro personajes.

En un aula del *Ramiro de Maeztu* encuentro a D. Francisco Suárez dando clases de Matemáticas a aquellos jóvenes aspirantes al Bachillerato Internacional. Jamás encontré igual vocación en este oficio: rigor, detalle, claridad, pasión en números y teoremas, curvas e infinitos.

Dejando a un lado las nobles Matemáticas, el arte de vivir se condensa en Daniel: Leonardo de Opañel, soñador de fin de milenio, espíritu libre. El es trotamundos, cuentista, portero de futbito, flautista, ciclista, mascota de Kentucky Fried Chicken, escalador, jugador de Rol, profesor moderno, Rambo, economista, cruzado de alcantarillas, mecenas, bombero y cantautor. Es además mi amigo.

Por último, Miguel y Marisa, con quienes es fácil descubrir que dar vale más y es más grato que recibir.

Por supuesto, quiero agradecer a mis padres su aliento desde siempre y su amor de padres. A mis hermanos, el estupendo ejercicio de sus funciones (las de hermano, claro) y al resto de mi pequeña familia, por su afecto perenne.

Y a Elena, por estar conmigo dando color a mi vida.

Armando Malanda Trigueros. Pamplona, 6 de Agosto de 1999.

RESUMEN

El intrincado problema del diseño de cuantificadores vectoriales, esto es, la obtención de librerías de código con las que la codificación de señales tenga las menores distorsiones posibles, se hace aún más complejo cuando son considerados los efectos del ruido en el canal, plasmados en la Tasa de Error por Bit (Bit Error Rate o BER).

Muchos de los algoritmos de diseño de cuantificadores vectoriales, entre los que destaca el GLA (Algoritmo de Lloyd Generalizado), no son capaces de sortear los numerosos mínimos locales subóptimos que presenta la función de distorsión media de la cuantificación en el espacio de las librerías de código. Por ello es preciso ejecutarlos repetidas veces, partiendo de puntos de inicio diferentes.

En esta Tesis se han querido explorar las posibilidades que brindan los Algoritmos Genéticos y otras técnicas heurísticas en el diseño óptimo de cuantificadores vectoriales sujetos a errores de canal.

Los Algoritmos Genéticos (AG) son procedimientos de optimización global iterativos y estocásticos inspirados en algunos mecanismos que rigen la dinámica de la Naturaleza, en particular la selección natural, la codificación genética y la reproducción heterosexual. Un AG contiene una población de individuos pertenecientes al espacio de posibles soluciones, que compiten entre sí y evolucionan tratando de maximizar alguna función de prestaciones o minimizar alguna función de coste definida sobre ese espacio. Esta evolución se basa en la selección de los mejores individuos y la eliminación de los peores, junto con diversos mecanismos para procrear nuevos individuos (hijos) a partir de los anteriormente seleccionados (padres).

Se plantean tres métodos distintos:

- El AGCV (Algoritmo Genético para la Cuantificación Vectorial): es un genético en el que los individuos de la población son tentativas librerías de vectores código. Para facilitar su evolución hacia puntos de mínima distorsión, se incorpora, como mecanismo de búsqueda local, el algoritmo GLA.

- El ARL (Algoritmo Refinado de Lloyd): es un algoritmo heurístico en el que se ejecuta sucesivas veces el algoritmo GLA, preservando siempre los mejores vectores código hallados hasta el momento. A medida que el algoritmo progresa, el número de vectores nuevos (no preservados) se va haciendo menor, con el objeto de que la búsqueda vaya siendo progresivamente más local.

- El AHCV (Algoritmo Híbrido para la Cuantificación Vectorial): es otro genético en el que se parte de una librería de códigos ya conocido, y lo que se optimiza es

la asignación de los códigos binarios disponibles, a los vectores código de la librería.

Los dos primeros son sometidos a extensas pruebas de simulación y contrastados con tres algoritmos de reputado nombre, corroborándose su adecuación al diseño de cuantificadores vectoriales. El tercero se plantea como una técnica posible, aún sin explorar ni probar exhaustivamente, que abre el camino a una nueva manera de utilizar los AG en este problema.

Al margen de estos métodos, una segunda cuestión abordada en esta Tesis es la reformulación de los principios de la Cuantificación Vectorial cuando las condiciones del canal no se suponen fijas o bien conocidas, sino que son descritas mediante la función densidad de probabilidad del BER. A este respecto se determina analíticamente la nueva función de distorsión y las reglas de optimalidad para el diseño de cuantificadores óptimos. Esto constituye un nuevo punto de partida para el diseño de cuantificadores vectoriales con planteamientos más realistas que los normalmente considerados.

ABSTRACT

The involved problem of Vector Quantization (VQ) design, i. e. the search for codebooks which yield as minimum distortions as possible, turns even more complicated when channel noise effects, characterised by the Bit Error Rate (BER), are considered.

Many VQ design techniques, included the most famous GLA (Generalized Lloyd Algorithm), are unable to avoid the sub-optimum local minima present in the quantification distortion function. Thus repeated executions, with different starting points are needed.

In this Thesis the possibilities offered by Genetic Algorithms and other heuristic techniques for noisy channel VQ design are explored.

Genetic Algorithms (GA) are stochastic and iterative global optimisation procedures, inspired on various mechanisms which rule Nature dynamics, such as natural selection, genetic coding and heterosexual reproduction. A GA contains a population of individuals belonging to the solution space, which compete each other and evolve towards maximisation of some performance function or minimisation of some cost function defined throughout this space. This evolution is carried out by means of selecting the fittest individuals in the population while removing the worst ones, as well as by several mechanisms for creating new individuals (offsprings) from selected ones (parents).

Three new methods are proposed:

- AGCV (Vector Quantization Genetic Algorithm): a GA in which individuals are tentative codebooks of the VQ scheme. To ease the evolution towards minimum distortion points, the GLA is included as a local search mechanism.

- ARL (Lloyd Refinement Algorithm): an heuristic algorithm in which GLA is run several times, preserving the best codevectors encountered so far. As the algorithm progresses, the number of codevectors removed and replaced by new ones is decreases, making the search progressively local.

- AHCV (Vector Quantization Hybrid Algorithm): another GA which starts from an initial fixed codebook and tries to optimise the assignment of the available binary codes to the vectors in the codebook.

The two first ones are submitted to extensive simulation tests and compared to three well-reputed methods in the field, confirming their adequacy to the problem under study. The third one is given as a possible technique, without having been exhaustively explored or tested so far; it only leads the way to a new manner of using GA for VQ design.

Apart from this, a second question faced in this Thesis is the reformulation of the VQ principles when channel conditions are not supposed fixed or well known, but are described by the probability density function of the BER.

To this respect, a new distortion function and optimality laws are analytically determined. This constitutes a new starting point for VQ design with more realistic basis than normally considered.

ÍNDICE

"Como de cada especie nacen muchos más individuos de los que pueden sobrevivir, y como, en consecuencia, hay una lucha por la vida, que se repite frecuentemente, se sigue que todo ser, si varía, por débilmente que sea, de algún modo provechoso para él bajo las complejas y a veces variables condiciones de la vida, tendrá mayor probabilidad de sobrevivir y de ser así naturalmente seleccionado."

Charles Darwin. "El Origen de las Especies", 1859.

Lena

Baboon

Pepper

Capítulo 1

INTRODUCCIÓN

La sociedad moderna, de forma más pronunciada año tras año, se organiza y evoluciona sobre la base de la disposición y uso de la información. Es difícil encontrar aspectos relevantes de la vida actual que no vengan determinados o, siquiera, condicionados por la posibilidad de disponer o enviar información. Sacar dinero del banco, matricularse de una asignatura, adquirir un pasaje de avión, pasar una revisión médica o simplemente sentarse en el sofá a ver la televisión o a hablar por teléfono son sólo ejemplos de la ineludible "sociedad de la información" que invade, fortaleciendo y moldeando la educación, la cultura de los pueblos, las relacionas de las personas, el trabajo, la sanidad y el ocio.

Los avances científicos de las tecnologías de la información y de las comunicaciones en las últimas décadas han posibilitado esta nueva situación del quehacer humano. A su vez, los nuevos hábitos y las posibilidades despertadas han impulsado el desarrollo tecnológico al demandar continuamente capacidades crecientes y precios siempre a la baja. Tal espiral ascendente (sistema con realimentación positiva) del desarrollo y de la demanda de las tecnologías de la información vienen generando un cambio que, por su celeridad, enraizamiento en la sociedad y carácter global, quizás podría acuñar el término de "revolución".

La transmisión y almacenamiento de masivas cantidadeds de imágenes, vídeo, voz y audio sobre el soporte de complejas redes de transmisión o de sofisticados sistemas de registro han venido de la mano del desarrollo tecnológico, tanto en los dispositivos y medios de transmisión, como en los algoritmos y arquitecturas de procesamiento de señales. Los primeros han logrado anchos de banda para la transmisión y capacidades de almacenamiento cada vez mayores; los segundos han proporcionado representaciones cada vez más eficientes de las señales.

1.1- Compresión de señales

En el cuadro anterior se ubica la compresión o codificación de señales, que persigue cuatro metas principales:

- minimizar la ocupación requerida para su transmisión por un canal,
- minimizar la capacidad necesaria para su almacenamiento,
- proporcionar una precisa y compacta descripción de las mismas para posteriores procesos como los de su encriptado, transmisión o clasificación, y por último,
- permitir su estandarización, requerimiento imprescindible para el libre crecimiento de la oferta de productos comerciales de amplia compatibilidad y para la implantación de la tecnología multimedia.

En la Tabla 1.1 tomada de [Bhaskaran 97] se muestran varias aplicaciones importantes para la transmisión o almacenamiento de imágenes, voz, audio o vídeo,

indicándose las capacidades requeridas para su representación antes y después de efectuarse la compresión.

Aplicación	Velocidad de transmisión (sin codificación)	Velocidad de transmisión (con codificación)
Voz (8 Kmuestras/seg. 8 bits/muestra)	64 Kbps	2-4 Kbps
Video de baja velocidad (tamaño de la trama 176x120, 24 bits/pixel)	5.07 Mbps	8-16 Kbps
Audioconferencia (8 Kmuestras/seg. 16 bits/muestra)	128 Kbps	6-64 Kbps
Videoconferencia (15 fps) (tamaño de la trama 352x240, 24 bits/pixel)	30.41 Mbps	64-768 Kbps
Audio digital (estéreo) (44.1 Kmuestras/s, 16 bits/muestra)	1.5 Mbps	128-768 Kbps
Transferencia de ficheros de video (tamaño de la trama 352x240, 24 bits/pixel)	30.41 Mps	384 Kbps
Video digital en CD (30 fps) (tamaño de la trama 352x240, 24 bits/pixel)	60.83 Mbps	1.5-4 Mbps
Video difusión (30 fps) (tamaño de la trama 720x480, 24 bits/pixel)	248.83 Mbps	3-8 Mbps
TV de alta definición (tamaño de la trama 1280x720, 24 bits/pixel)	1.33 Gbps	20 Mbps

Tabla 1.1

Estos tipos de señales son apropiadas para la compresión debido fundamentalmente a su elevada redundancia, bien espacial, entre pixeles adyacentes de una imagen, bien temporal, entre muestras contiguas de voz o audio e incluso espacio temporal, entre imágenes consecutivas de vídeo o TV. Por otro lado, buena parte de la información contenida en ellas es irrelevante desde un punto de vista perceptual, lo cual constituye otra forma distinta de redundancia. La compresión perseguirá eliminar estos tipos de redundancia, dando a las señales representaciones más compactas.

El proceso de compresión puede ser descrito según la Figura 1.1

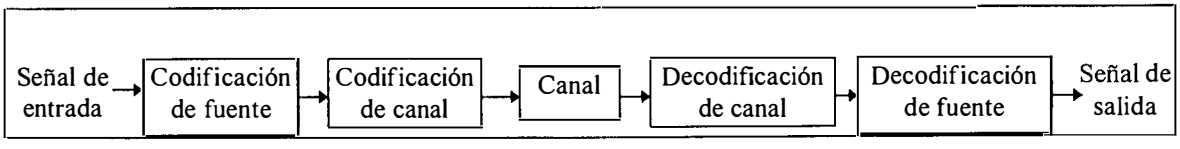

Fig 1.1 Proceso de compresión de una señal

El codificador de fuente lleva a cabo el proceso de compresión, reduciendo la tasa de datos de entrada según el índice de compresión, que indica la relación entre el tamaño de los datos a su salida y a su entrada. Después del codificador de fuente suele haber otro bloque de codificación dedicado a transformar la señal comprimida en otra representación adecuada para la transmisión fiable a través de un canal ruidoso. A este proceso se le da también el nombre de codificación de canal o de control de errores.

En sus desarrollos sobre la Teoría de la Información, Shannon dedujo límites teóricos para la transmisión de información en canales de banda limitada y para la compresión de datos [Shannon 48], [Shannon 59]. Asimismo mostró que se podían construir sistemas de comunicación prácticamente óptimos diseñando por separado el codificador de fuente y el de canal, aunque su Teoría no recogía técnicas explícitas para diseñar tales codificadores.

Por esta razón y por ser menor la complejidad planteada en el diseño, en la mayor parte de los sistemas, ambas codificaciones son concebidas como bloques independienes, lo cual no implica necesariamente que su comportamiento se aproxime al óptimo teórico. Es más, el diseño conjunto de codificadores de fuente y de canal está cobrando creciente importancia en los últimos años [Farvardin 87], [Liu 93] [Skinnemoen 94] .

Siguiendo el flujo de la información en la Figura 1.1, la señal a la salida del segundo codificador es enviada por un canal de transmisión. A la salida de éste, es sometida a sendos procesos de decodificación, inversos a los de codificación de canal y de fuente.

Desde un punto de vista general, el diseño de un sistema de codificación se plantea como un problema de minimización conjunta de varias variables:

- Velocidad de transmisión: tasa de bits/seg entregados al canal. Este parámetro suele estar limitado por la entropía (contenido de información de la fuente), en el sentido de que fuentes con mayores entropías son más difíciles de comprimir.

- Complejidad en la implementación: requerimientos computacionales que precisa el codificador y el decodificador, medidas normalmente en instrucciones en punto fijo o en punto flotante por segundo. También hay que incluir en este punto la cantidad de memoria necesaria. La trascendencia de la complejidad en la implementación puede ser muy diferente en el codificador y en el decodificador. Es el caso de los sistemas de difusión, en los que un codificador envía información a un gran conjunto de decodificadores.

- Complejidad en el diseño: a veces los cálculos en el diseño son tan elevados y los conjuntos de prueba tan extensos que su reducción puede redundar en un considerable ahorro de tiempo y dinero.

- Retardo neto: un proceso complejo de compresión frecuentemente conlleva retardos considerables en la transmisión y en la recepción. Algunas aplicaciones, sobre todo las que incluyen interactividad, pueden llegar a degradarse muy perceptiblemente por causa de estos retardos. A veces pueden ser reducidos aumentando la potencia del procesador. Cuando esto no es posible se suele recurrir a codificaciones menos sofisticadas, quizás en detrimento de otras figuras de mérito como la calidad o la velocidad de transmisión.

- Calidad de la compresión: esta figura, que tiene sentido en los codificadores con pérdidas, suele tomarse como la relación señal a ruido, donde este último se toma como la diferencia entre la señal de salida y la de entrada. En el caso de imágenes o vídeo, se utiliza normalmente la relación señal de pico a ruido, donde por "señal de pico" se entiende la potencia máxima posible de la imágenes codificada con un número determinado de bits. Estas dos magnitudes revelan aspectos cuantitativos de las señales, pero muchas veces, para evaluar la calidad de un sistema, es preciso incluir pruebas de percepción en las que un conjunto de personas las valoren subjetivamente.

Las figuras de mérito mencionadas presentan distinto grado de importancia en diversas aplicaciones. Normalmente en el diseño se busca minimizar una de ellas, estableciendo "a priori" unos rangos permitidos o valores concretos para el resto, frecuentemente impuestos por condiciones del diseño no sujetas a modificación (procesador empleado, ancho de banda asignado, etc.). Incluso a veces ocurre que alguno de los anteriores factores es irrelevante (retardo neto en sistemas de difusión, complejidad en el diseño en algunos casos, etc.).

1.1.1- Clasificación de métodos de compresión

Una clasificación taxonómica de los métodos de compresión existentes tomada de [Bhaskaran 97] es la de la Figura 1.2.

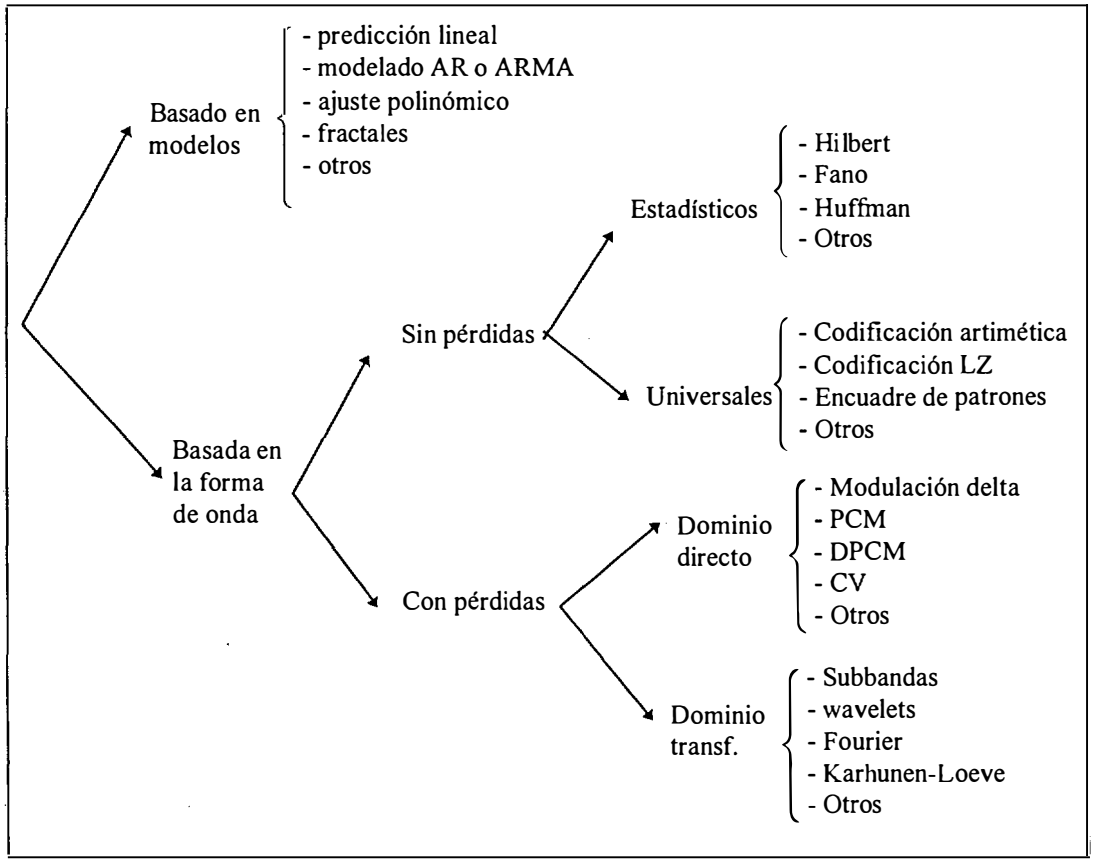

Fig. 1.2 Métodos de compresión de señales

Otras clasificaciones, atendiendo a los criterios planteados a continuación, pueden ser:

a- Tipo de reconstrucción de la señal:

a.1- Sin pérdidas: la señal a la salida del codificador es exactamente igual a la de entrada. Obviamente en estos sistemas la calidad es infinita y no está sujeta a minimización. Normalmente la compresión de señales e imágenes médicas, dadas sus delicadas implicaciones, se incluyen en esta categoría.

a.2- Con pérdidas: se permite cierta degradación en la señal reconstruida, lo cual hacce posible aumentar la eficacia del codificador (reducir la velocidad de transmisión). La mayor parte de las aplicaciones en imagen, vídeo, voz y audio pertenecen a este grupo, pues no es necesario que la señal sea exactamente idéntica a la señal de entrada.

b- Tasa de bits:

b.1- constante: generalmente empleado cuando las capacidades de transmisión o almacenamiento son fijas.

b.2- cuando los requerimientos de transmisión o almacenamiento no son fijos y sí lo es la calidad del sistema, se pueden asignar de forma variable distinto número de bits a los sucesivos grupos de muestras.

c- Tipo de compresión:

c.1- Simple: se procesan y comprimen una a una las muestras de señal o píxeles que llegan a la entrada.

c.2- Múltiple: se procesan a la vez varias muestras de entrada consecutivas o píxeles adyacentes. A la salida se entrega de una vez la representación codificada de todo el conjunto. También a estos sistemas se les llama codificadores de bloque.

d- Dominio

d.1- Directo: el proceso de compresión tiene lugar sobre las muestras tomadas como función del tiempo, en señales de voz o audio, de la posición, en el caso de imágenes, o de la combinación tiempo-posición en el caso de señales de vídeo.

d.2- Transformado: el proceso de compresión se lleva a cabo sobre la señal previamente transformada. El nuevo dominio puede ser la frecuencia, la combinación tiempo-frecuencia, el dominio wavelet, fractal, etc.

1.1.2- Cuantificación Vectorial

En su Teoría de la Información, Shannon propuso un tipo muy particular de codificador de fuente múltiple, que asignaba bloques de muestras consecutivas no solapadas de la señal de entrada en símbolos de salida binarios siguiendo una regla específica, sin tener en cuenta las acciones anteriores del sistema (codificación sin memoria con respecto a los bloques de entrada). El decodificador, a su vez, asignaba los símbolos binarios que le llegaban a vectores de muestras de salida (vectores representantes de los bloques de muestras de entrada).

Para una fuente de señal dada y un conjunto de vectores representantes, el criterio de optimalidad que Shannon proponía era el de minimización de la distorsión media, para lo cual era preciso que el codificador operara siguiendo la consigna del Vecino Más Próximo. En lenguaje natural esta regla se puede formular como sigue: dado un vector de entrada, el codificador ha de seleccionar el código binario que, una vez decodificado, genere a la salida el vector representante on menor distorsión con respecto al de entrada, entre todos los representantes existentes.

Shannon llamó a este sistema "codificación de fuente sometida a criterio de fidelidad". Más adelante se le comenzó a llamar Cuantificación Vectorial (CV), denominación que ha prevalecido desde entonces.

Además Shannon presentaba las reglas que debían cumplir los codificadores óptimos, pero no proponía métodos específicos para hallarlos. Este asunto, a saber, la obtención de un conjunto óptimo de vectores representantes en un sistema de Cuantificación Vectorial constituye el objeto de la presente Tesis.

1.2- Optimización

1.2.1- Panorámica de problemas y técnicas

Antes de profundizar en la revisión de las técnicas existentes y de las aquí propuestas para el diseño de cuantificadores vectoriales óptimos, conviene reparar, si quiera de manera superficial, sobre el hecho en sí de la optimización, sus metas y los distintos planteamientos metodológicos existentes.

La optimización se puede describir de manera general como el problema de encontrar los valores de ciertas variables incógnitas del sistema sometido a análisis, para hacer mínima una función de coste (o máxima una función objetivo), con la posible imposición de diversas restricciones. Formalmente, se trata de encontrar $\bar{x}_{min} \in S$ tal que,

$$f(\bar{x}_{min}) \leq f(\bar{x}), \qquad \forall \bar{x} \in S \qquad \textbf{(1.2.1.1.a)}$$

y sujeto a:

$$\begin{aligned} g_i(\bar{x}) &= \bar{a}_i \qquad i = 1, 2, \ldots, N \\ h_j(\bar{x}) &\leq \bar{b}_j \qquad j = 1, 2, \ldots, M \end{aligned} \qquad \textbf{(1.2.1.1.b)}$$

siendo $g_i(\bar{x})$ y $h_j(\bar{x})$ funciones específicas del problema, y $\bar{a}_i$ y $\bar{b}_i$ constantes del mismo.

La mayor parte de los problemas encontrados en las diversas disciplinas de la Ingeniería y de algunas otras ramas del saber como la Economía, la Sociología, etc. pueden formularse de la manera anterior; esto es, son en esencia problemas de optimización. No es de extrañar que se hayan desarrollado numerosísimas técnicas de ámbito general o específico que intentan optimizar.

Para aportar una visión de conjunto, un marco general sobre la teoría y práctica de esta disciplina, quizás arte de la Optimización parece sensato comenzar por la clasificación, tanto de los problemas que trata de resolver y del tipo de soluciones buscadas, como de los métodos con que cuenta.

a- Atendiendo a la tipología del problema, aparecen varios casos particulares de la propuesta general definida según (1.2.1.1) y que presentan un notable interés práctico:

a.1- Optimización lineal: las funciones $f(\bar{x})$, $g_i(\bar{x})$ y $h_j(\bar{x})$ son funciones lineales en $\bar{x}$. Este tipo de problemas se encuentra frecuentemente en la asignación de recursos y organización de operaciones;

a.2- Optimización cuadrática: la función objetivo $f(\bar{x})$ es cuadrática en la variable $\bar{x}$. Dado que la potencia y la energía suelen proceder de otras variables más fundamentales, elevadas al cuadrado (diferencia de tensión, intensidad de campo, amplitud de las muestras de una señal, etc.)(*), esta categoría alberga a gran cantidad de problemas en los que se busca aumentar la potencia o el rendimiento de un sistema, o disminuir el ruido presente.

a.3- Optimización combinatoria: el conjunto S de soluciones es numerable (ya sea finito o infinito). Más en particular, las posibles soluciones $\bar{x}$ pueden expresarse como combinaciones, variaciones o permutaciones de uno o varios conjuntos numerables. Pertenecen a este grupo todos aquellos problemas que tratan de establecer un ordenamiento o una clasificación para alcanzar el máximo aprovechamiento de algún recurso limitado. Es por ello que se pueden encontrar abundantes ejemplos en actividades de diseño tan diversas como la planificación urbanística, el diseño de circuitos, la asignación de radiofrecuencias, la gestión de redes de datos o la organización de operaciones en plantas industriales.

Mención aparte merecen algunos problemas teóricos, aunque de gran interés práctico, como el *Problema del Viajante de Comercio* (*Salesman Travel Problem*) [Díaz 96], el *Problema de la Mochila* (*Knapsack Problem*) **[Díaz 96]**, el *Dilema del Prisionero* (*Prisioner's Dilemma*) [Goldberg 89]. Traídos de la Teoría de Juegos o de la Investigación Operativa presentan, junto a un sencillo enunciado, una elevada complejidad y constituyen auténticos bancos de prueba en los que se contrastan algoritmos de optimización combinatoria.

a.4- Optimización sin restricciones: la parte b) de la ecuación (1.2.1.1) no existe y por tanto basta con encontrar la mejor solución de S. Aunque suelen ser menos frecuentes en la práctica, este tipo de problemas también tiene una difusión muy extensa en las distintas disciplinas técnicas. Es más, se podrían reformular todos los problemas de optimización para que cupieran en esta categoría. Bastaría con incluir las restricciones en la definición de S. Sin embargo, es más usual recoger las dos alternativas y formular la optimización con o sin restricciones explícitas.

b- En cuanto al tipo de soluciones buscadas, cabe el planteamiento de empeñarse por encontrar una solución exacta que optimice a $f(\bar{x})$, o sólo tratar de buscar una "buena solución". La gran mayoría de los métodos de optimización converge hacia soluciones localmente óptimas (pudiendo quizás oscilar en torno a ese óptimo o acabar en un punto próximo a él). Sin embargo, gran cantidad de problemas prácticos presentan funciones de coste (o funciones objetivo, si se refieren a maximizaciones) multipolares. Garantizar que la solución hallada es el óptimo global, que o se encuentra muy próxima

(*) Se refiere esta sentencia a que las aludidas variables aparecen como conceptos más elementales y primitivos en sus respectivos campos de la Física o de la Ingeniería donde residen. Generalmente se introducen antes, apareciendo la potencia y la energía como nuevas variables dependientes de aquéllas. Por ejemplo, raro sería encontrar primero el concepto de energía cinética y más tarde el de velocidad. No obstante, carece de sentido entrar aquí en la discusión de qué entendemos por más fundamental o por qué consideramos a estas últimas más fundamentales que las anteriores.

a él, resulta generalmente una cuestión inabordable con tiempos y capacidades de cómputo o de memoria limitados. Sin embargo, puede no resultar imprescindible, a veces ni si quiera ventajoso, llegar a la mejor solución, sobre todo cuando [Díaz 96]:

- los rangos posibles de mejora en la solución no permiten beneficios substanciales;
- los datos son poco fiables o el modelo es sólo aproximado. En este caso la solución encontrada será a la fuerza una aproximación de la solución real;
- la optimización sea sólo un paso intermedio para la aplicación de otro algoritmo desde un punto de partida adecuado;

En estas situaciones es preferible buscar sólo soluciones que sean mejores que la mayor parte de las posibles, incluidas otros óptimos locales.

c- Finalmente, para estructurar la gran variedad de métodos de optimización se propone la clasificación taxonómica de la Figura 1.3. En ésta se distingue en primer lugar los métodos directos de los iterativos. Los primeros, que buscan una solución analítica precisa de forma cerrada, están muy limitados a problemas sencillos. Más aún, a menudo ni siquiera resultan computacionalmente rentables frente a los segundos, cuya búsqueda se realiza paso a paso (piénsese, por ejemplo, en la optimización cuadrática, en la que hay que calcular matrices inversas o pseudoinversas; es común realizar estas operaciones de forma recursiva [Chong 96].

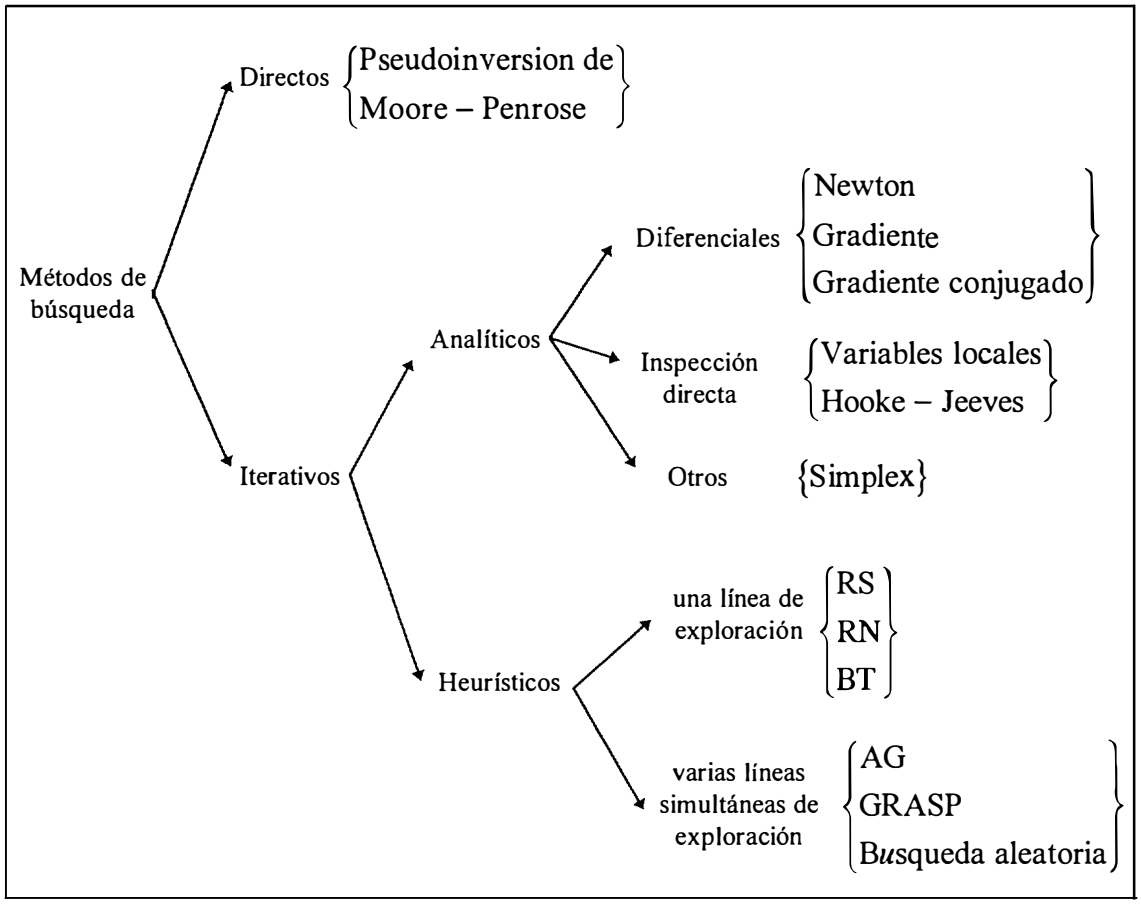

Fig.1.3 Clasificación de métodos de optimización

Centrándonos en los métodos iterativos, es habitual la distinción entre métodos analíticos y heurísticos. En los primeros la justificación y modo de funcionamiento tienen un claro soporte matemático. Estos, a su vez, pueden dividirse en métodos de búsqueda directa, que sólo utilizan en el proceso valores de la función (p.ej el método de las variables locales o el de Hooke-Jeeves [Michavilla 85]); métodos diferenciales, que utilizan además de los valores de la función, los de sus derivadas parciales (p .ej. los métodos del gradiente, del gradiente conjugado y de Newton [Michavilla 85]); y otros métodos que no pueden considerarse dentro de ninguna de estas dos subclases, como es el caso de *Simplex*, que resuelve problemas de optimización lineal [Chong 96].

La definición que da Zanakis para los métodos heurísticos es la de "procedimientos simples, a menudo basados en el sentido común, que se supone ofrecerán una buena solución (aunque no necesariamente la óptima) a problemas difíciles, de un modo fácil y rápido" [Zanakis 81]. Estos aún podrían subdividirse en aquéllos que siguen una sola línea exploratoria, esto es, que un punto inspeccionado conduce al siguiente y así sucesivamente hasta el final de la búsqueda, y aquéllos que siguen a la vez varios caminos de exploración. Entre los primeros destacan el Temple Simulado, las Redes Neuronales y la Búsqueda Tabú. Entre los segundos cabe mencionar a los Algoritmos Genéticos y a la Búsqueda Grasp [Díaz 96].

1.2.2- Técnicas heurísticas

Aunque el término procede de la palabra griega *heuriskein* que significa encontrar o descubrir, estos procedimientos sólo pueden considerarse procesos de búsqueda que, como se apuntaba en la definición del apartado anterior, suelen finalizar en una solución subóptima.

En los últimos 20 años, el crecimiento exponencial de las capacidades computacionales ha permitido a estas técnicas desbancar a los tradicionales métodos analíticos en muchas ocasiones y complementarlas en otras, y también afrontar problemas cuya solución parecía infranqueable anteriormente.

Todo ello ha conducido, a su vez, a la aparición de nuevos algoritmos de búsqueda y a un importante desarrollo en el campo de la complejidad computacional.

En lo que sigue se describen de forma introductoria los heurísticos antes enumerados.

- El Temple Simulado (Simulated Annealing) hace uso de los conceptos de la Mecánica Estadística, en particular, trata de emular el proceso térmico para obtener estados de baja energía en un sólido. En este proceso primero se reblandece el sólido mediante calentamiento y luego se va enfriando lentamente para que las partículas vayan asentándose en estados de mínima energía. Al pasar lentamente de una temperatura a otra más baja, el sólido va pasando por sucesivos estados de equilibrio térmico, hasta llegar al estado final de mínima energía. Si el enfriamiento es rápido, los niveles no corresponden a los de mínima energía y el sólido presentará imperfecciones.

Kirkpatrick trasladó este concepto al campo de la optimización [Kirkpatrick 83] haciendo corresponder las soluciones tentativas del problema y la función de coste, con estados diferentes del sistema y con su energía, respectivamente. Además se utilizaba un parámetro de control que hacía las veces de temperatura del sistema.

Partiendo de una solución inicial tomada al azar, se contemplan desplazamientos locales de esta solución. Siempre que el nuevo punto inspeccionado presente menor energía que su predecesor, será elegido como nuevo candidato, en caso contrario, también podrá ser elegido aunque sólo con una cierta probabilidad, función del parámetro de control. Para un determinado valor de este parámetro, la solución converge hacia un estado de equilibrio. Haciendo descender este parámetro de control (temperatura) se va pasando por distintos estados de equilibrio hasta llegar a una solución de energía mínima en el caso ideal (con un número infinito de puntos de equilibrio intermedios) y sólo "pequeña" en el caso real.

- Los Algoritmos Genéticos [Holland 75] son procedimientos de búsqueda inspirados en la evolución natural y en la Genética. En ellos siempre existe una población de individuos que son soluciones posibles al problema de optimización. Dichos individuos se representan (codifican) de manera que sea sencilla la unión de dos de ellos (padres) para la obtención de uno nuevo (hijo), próxima solución tentativa del problema.

Al igual que la selección natural y el traspaso genético de padres a hijos son los motores fundamentales de la evolución de las especies, en los AG, la elección de individuos con menores funciones de coste (más adaptados) y el cruce entre parejas de individuos van imponiendo el progreso de la población, conduciéndola hacia soluciones cada vez mejores.

Además es común incorporar un mecanismo que se aplica a los individuos esporádicamente y que altera alguna de sus características (desplaza la solución representada). Este mecanismo recibe el nombre de mutación, por emular el proceso de la mutación genética y contribuye de forma importante en el proceso de búsqueda, por abrir nuevas rutas de inspección, al igual que en la Naturaleza ha posibilitado la aparición de nuevas razas y especies.

- La Búsqueda Tabú, inicialmente propuesta por Glover [Glover 86], es un procedimiento para guiar a un mecanismo de búsqueda local, intentando evitar la caída en óptimos locales. Esta búsqueda selecciona el mejor de los movimientos posibles en cada paso, aunque permite el movimiento a soluciones peores que la actual, pudiendo así escapar de la atracción de óptimos locales no globales. Para evitar que el proceso vuelva a un viejo óptimo local, la búsqueda clasifica un determinado número de los más recientes movimientos como "movimientos tabú", que no pueden realizarse en un determinado intervalo de tiempo.

- La Técnica GRASP: del Inglés (Greedy Randomized Adaptive Search Procedure) es un procedimiento formulado por Feo y Rinde para solucionar problemas de alta complejidad [Feo 89]. Consta de dos fases, la de construcción y la de búsqueda local. En la primera se busca una solución tentativa de forma iterativa, eligiendo un elemento a cada paso. Este elemento se elige al azar dentro de una lista restringida de candidatos cuyas limitaciones aseguran que se vaya construyendo una solución buena, aunque no necesariamente óptima.

La búsqueda local puede llevarse a cabo con técnicas tipo gradiente o de cualquier otro tipo. La idea global de GRASP es producir una diversidad de buenas soluciones esperando que al menos una de ellas esté en un entorno de la solución óptima o en el de una solución de valor cercano al óptimo.

- Las Redes Neuronales artificiales son sistemas paralelos y distribuidos capaces de almacenar conocimiento experimental adquirido a base de ejemplos [Haykin 94]. Su estructura y funcionamiento básico están, en cierto sentido, inspiradas en los mecanismos cerebrales, en particular en las conexiones entre neuronas y en su capacidad para almacenar y transmitir información de forma distribuida.

El elemento básico de proceso de una red neuronal es la neurona que presenta varias entradas y una salida. Los datos que aparecen a sus entradas son multiplicados por sendas constantes (sinapsis o pesos de la red), sumados y sometidos a una operación no lineal antes de ser entregados a su salida.

La red presenta varias neuronas conectadas entre sí de forma diversa y unida al exterior a través de varias entradas y salidas. Globalmente se comporta como un sistema no lineal dependiente de los valores que tengan los pesos de la red. El funcionamiento de la red se divide en dos fases: el entrenamiento y la ejecución propiamente dicha.

En la primera fase se presenta a la red una colección de ejemplos que representen estadísticamente a la señal de entrada. Mediante un mecanismo interno, la red va modificando sus pesos en el sentido de hacer mínima alguna función de coste asociada con los datos de salida de la red. Este proceso se asemeja al de aprendizaje, por lo que también recibe este nombre.

En la fase de ejecución, la red responde a las entradas según su estructura original y los valores de los pesos fijados durante el aprendizaje. En esta segunda fase los pesos no varían.

Aunque especialmente capacitadas para asumir tareas complejas de reconocimiento, clasificación y control, algunas de sus variantes, como la red de Hopfield, pueden ser usadas como mecanismos globales de optimización global. Otras, como la de Kohonen pueden utilizarse para la optimización de algunos tipos particulares de problemas combinatorios [Díaz 96].

1.3- Motivación, objetivos y estructura de la Tesis

Habida cuenta de la importancia creciente del problema de compresión de señales y de la complejidad del diseño de estos sistemas, en particular en los cuantificadores vectoriales, sobre todo al considerar los efectos debidos al ruido de canal, en esta Tesis se ha querido explorar las posibilidades que brindan los AG en el diseño de estos sistemas. Al respecto se plantean dos procedimientos distintos, cuyo esqueleto principal es un AG adaptado al problema. Su enfoque es diferente, como también lo es el detenimiento y esfuerzo empleados en uno y otro: el primero (el AGCV) se presenta estudiado en detalle, justificada la selección del valor de cada parámetro y exhaustivamente probado. El segundo (el AHCV) se expone como una técnica posible, sin pulir ni probar de forma sistemática (ni siquiera los resultados publicados en [Malanda 98] se presentan aquí). Sólo se pretende definir esta nueva alternativa, ya abierta.

A la par que estas técnicas genéticas, también se incluye en este trabajo otro algoritmo heurístico para optimizar CV (el ARL). Es éste un nuevo método específicamente diseñado para resolver el problema y que no se enmarca en ninguno de las corrientes heurísticas antes señaladas. Se incluye en esta Tesis por constituir una alternativa diferente, complementaria y novedosa a las técnicas genéticas propuestas.

Más concretamente, el objetivo principal de esta Tesis es mostrar la conveniencia de los métodos AGCV y ARL para la obtención de librerías de código óptimas en la CV cuando las condiciones del ruido en el canal son conocidas y están dadas por su tasa de error de un bit.

Tras este capítulo introductorio, la Tesis se estructura de la siguiente manera. En el Capítulo 2 se revisan los fundamentos; por un lado se repasan los principios de la CV, desglosándola en dos partes, que hacen referencia a la presencia o ausencia de errores en el canal. Muy en particular, se detallan dos condiciones necesarias para el diseño de cuantificadores óptimos: las Reglas de los Centroides y del Vecino Más Próximo, ambas con versiones diferenciadas para las dos situaciones anteriores.

Por otro lado, se explican, con intención panorámica, a la vez que con cierto grado de detalle, las Técnicas Evolutivas, conjunto de métodos globales de optimización inspiradas en el mecanismo de Selección Natural de las Especies, entre las que destacan los Algoritmos Genéticos.

En el Capítulo 3 se revisa el *estado del arte*, nuevamente separando el problema en los dos planteamientos anteriores (con o sin errores de canal). Para cada uno de ellos, se presenta una colección diversa de técnicas, muchas de ellas, particularizaciones al problema tratado de métodos más generales de optimización global. Es el caso de los Algoritmos Genéticos, el Temple Simulado, las Redes Neuronales o los Métodos Fuzzy. Otros, en cambio, son propuestas específicas para la optimización de CV.

En la optimización de CV en presencia de errores de canal, las mencionadas técnicas se dividen en tres grandes grupos:

- aquéllas que optimizan la asignación de los índices binarios con que va a ser codificada una librería de códigos previamente diseñada (generalmente optimizada para el caso de transmisión sin error);
- aquéllas que llevan a cabo una optimización conjunta de vectores e índices, y
- aquéllas que realizan una optimización alternada e iterativa de vectores e índices.

En el siguiente capítulo se describen las técnicas algorítmicas pospuestas en esta Tesis. Según se comentaba más arriba, las dos primeras, el AGCV e el ARL se explican en detalle, mostrándose sus diversas alternativas, así como las pruebas realizadas para determinar las más adecuadas y para ajustar sus parámetros. El AHCV se explica a un nivel más conceptual, sin llegar a determinar una configuración específicamente apropiada al problema, como se hace con los otros dos métodos

El Capítulo 5 está dedicado a los experimentos: ensayos de la cuantificación vectorial de imágenes digitalizadas bajo distintas condiciones de ruido de canal. Se muestran comparaciones del AGCV y el ARL con tres técnicas de reconocido prestigio en este campo y se extraen los resultados más relevantes.

A continuación (Capítulo 6) se abren disquisiciones y expresan comentarios sobre algunos aspectos particulares del trabajo realizado, en particular, sobre los métodos planteados, los resultados obtenidos en las simulaciones y el propio planteamiento del problema.

Es este último punto el que abre paso a la Adenda, en el Capítulo 7, en la que el problema se reformula ligeramente, considerando que ahora los errores en el canal no están dados por una tasa de error por bit (BER) fija y conocida, sino por un BER variable y sólo descrito probabilísticamente, en términos de su función densidad de probabilidad. A partir de esto, se extienden para este nuevo caso, las expresiones para

calcular la distorsión total del proceso CV, así como las condiciones de optimalidad para el diseño de cuantificadores. Además se recogen algunos casos simplificados y un sencillo ejemplo que muestra la diferencia de éste con el primer planteamiento.

En el último capítulo se extraen las conclusiones finales del trabajo, presentándose los aspectos aún abiertos para más detallado estudio, así como nuevas propuestas para futuras investigaciones.

Capítulo 2

FUNDAMENTOS

2.1- Cuantificación Vectorial

2.1.1- Introducción

La cuantificación surge en aquellas situaciones en las que una variable de una o varias dimensiones, que puede tomar valores dentro de un conjunto de cardinal finito o infinito, quiere ser expresada por medio de un segundo conjunto, finito, de cardinal inferior al primero. En la digitalización de señales analógicas el proceso de cuantificación aparece de forma natural, pues variables con rangos continuos han de ser representadas por medio de un número limitado de símbolos.

Cuando es preciso cuantificar variables multidimensionales, cada una de sus componentes puede someterse por separado al proceso de Cuantificación Escalar (CE). Sin embargo, cuando estas componentes contienen cierto grado de relación unas y otras, resulta más interesante, en términos de eficiencia de la compresión, cuantificarlas conjuntamente. La Cuantificación Vectorial (CV) se presenta así, como la generalización natural de la CE de una a varias dimensiones. Gran parte de las ideas, técnicas y algoritmos de diseño o implementación relativas a la CV tienen su origen en sus equivalentes unidimensionales, aunque en otros muchos casos, esto no ocurre así.

Por otra parte, mientras que la CE es, sobre todo, un proceso unido a la digitalización de señales analógicas, la CV está estrechamente ligada a la compresión de señales digitales, ya sean de naturaleza originaria analógica o digital. Por otra parte, también es frecuente el uso de la CV en aplicaciones de reconocimiento de patrones y clasificación.

Es el caso más frecuente que las componentes de un vector sujeto al proceso de CV sean las muestras consecutivas de una señal unidimensional (p. ej. señales de voz o audio) o de dos o incluso tres dimensiones (p. ej. imágenes o vídeo). La CV de voz, audio, imagen y vídeo ha tenido un importante crecimiento en los últimos años, participando en mayor o menor medida en diversos estándares de compresión de estas señales [Bhaskaran 97], [Rao 96]. Otros tipos de señales como los electrocardiogramas también han sido sometidas a compresión por medio de CV [Mammen 90], [Wang 97].

En los siguientes apartados de este capítulo se revisará por separado la CE, la CV sin errores de canal y la CV con errores de canal.

2.1.2- Cuantificación Escalar

2.1.2.1- Planteamiento

Mediante el muestreo, una señal analógica de una dimensión x(t) es convertida en la secuencia x[n] (señal en tiempo discreto) obtenida como la sucesión de los valores de x(t) en los instante seleccionados $\{..t_{-1},t_0,t_1,..\}$. Formalmente,

$$x[n] = x(t_n),\ n=\cdot\cdot,-1,\ 0,\ 1,\ \cdot\cdot. \qquad \textbf{(2.1.2.1.1)}$$

Tras el muestreo, el proceso de cuantificación asigna a cada muestra de x[n] un único valor $\hat{x}[n]$ perteneciente al conjunto finito $C = \{c_0,c_1,\ ,c_{L-1}\}$ de valores de cuantificación, de acuerdo con la regla de cuantificación (Figuras 2.1 y 2.2):

$$\hat{x}[n]= Q(x[n]) \qquad \textbf{(2.1.2.1.2)}$$

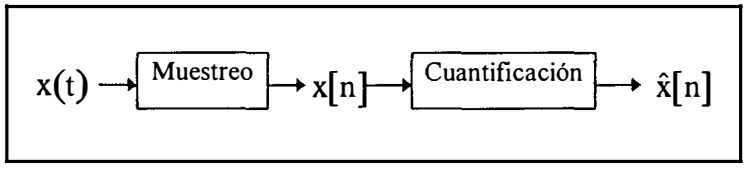

Figura 2.1. Muestreo y cuantificación de una señal analógica

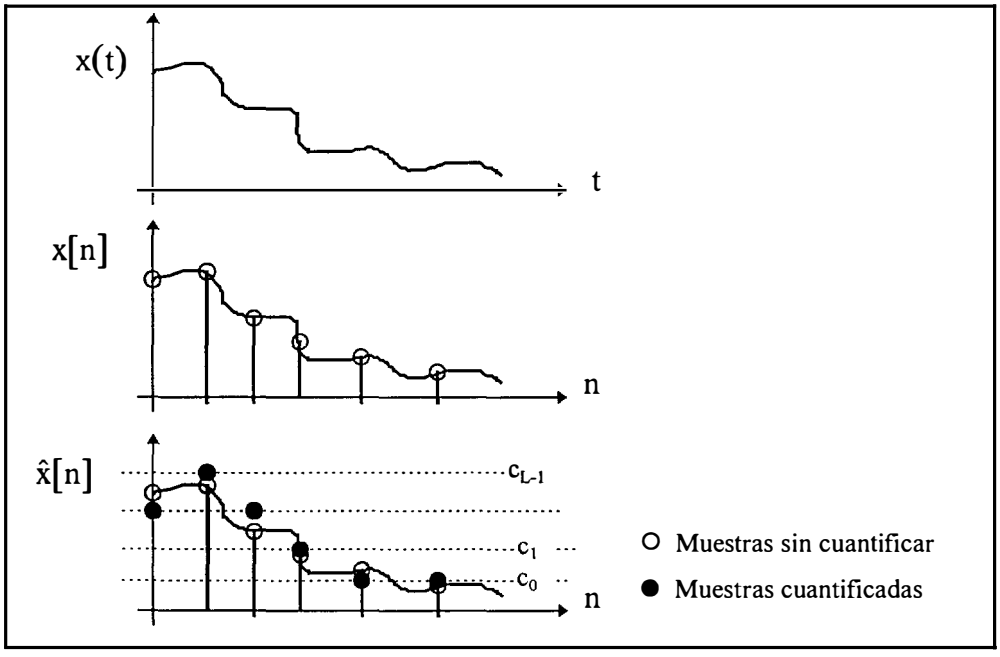

Fig. 2.2 Señales analógica x(t), muestreada x[n] y cuantificada $\hat{x}[n]$.

Así se consigue expresar cualquier valor de la variable x con un alfabeto de dimensión L, que a su vez puede representarse por medio de $\log_2 L$ bits ó $\log_M L$ símbolos M-arios.

Conjuntamente la cuantificación engloba los procesos de codificación y decodificación tal y como se muestra en la Figura 2.3.

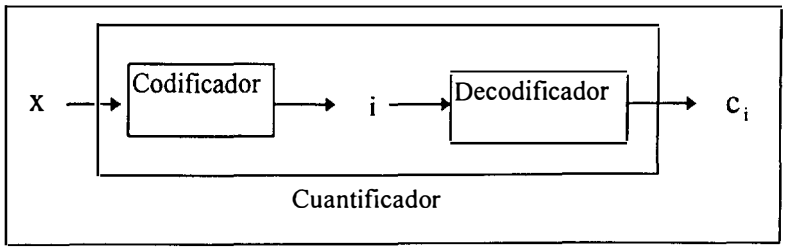

Fig. 2.3 Diagrama funcional de un cuantificador vectorial

En la primera parte del proceso, al vector x perteneciente al conjunto S de todos los valores que pueden llegar a la entrada, se le asigna el índice i, entre 0 y L-1, de acuerdo con una determinada regla de codificación:

$$Cod(x)=i \qquad i \in \{0,\ 1,\ ..,\ L\ \ 1\} \qquad \textbf{(2.1.2.1.3)}$$

Esta regla es la que define al cuantificador y conlleva implícitamente una partición del espacio S en subespacios S_i formados por los vectores de S que son codificados con los distintos índices i:

$$S_i = \{x \in S \,/\, Cod(x) = i\} \quad i=0,1,\ L\text{-}1. \qquad \textbf{(2.1.2.1.4)}$$

Sin pérdida de generalidad puede pensarse que los intervalos S_i son crecientes (ocupan porciones cada vez más a la derecha de la recta real). De esta forma, llamando x_i a sus puntos extremos, estos intervalos pueden expresarse como $S_i = (x_i, x_{i+1}]$ (*) (Figura 2.4). El espacio completo de muestras de entrada será el intervalo $S = (x_0, x_L]$.

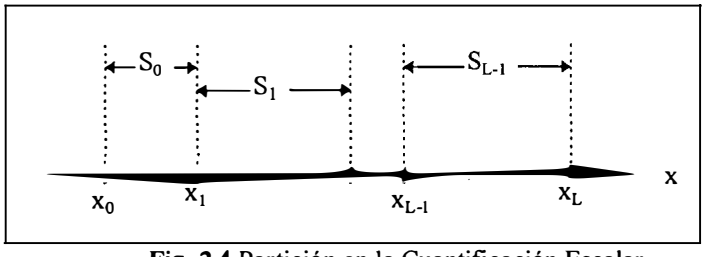

Fig. 2.4 Partición en la Cuantificación Escalar

Este proceso de codificación es irreversible, en tanto en cuanto, el número de elementos posibles de S sea mayor que L.

En la segunda parte de la cuantificación, al índice i se le asigna el vector c_i del conjunto C:

$$Decod(i) = c_i, \quad c_i \in C \qquad \textbf{(2.1.2.1.5)}$$

Al contrario que la codificación, éste sí que es un proceso reversible, ya que cada c_i tiene su índice i y viceversa.

(*) Por conveniencia notacional hemos dispuesto que todos los intervalos S_i sean abiertos en su extremo inferior y cerrado en el superior; en vez de este criterio, podría haberse tomado otro diferente, con tal de que el conjunto de los S_i formara una partición de S.

El cuantificador se define como regular si $c_i \in S_i$, lo cual normalmente se cumple, pues de no ser así, existiría un sesgo intrínseco en el proceso. En este caso, si los intervalos S_i son crecientes, también los valores de cuantificación $c_0, c_1, \ldots, c_{L-1}$ formarán una sucesión creciente, como aparecen en la Figura 2.5.

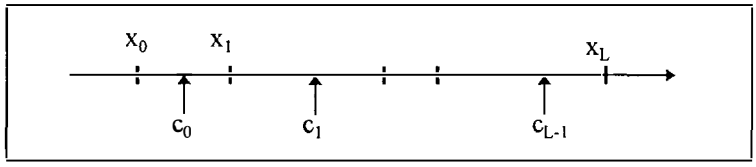

Fig. 2.5 Partición y valores representantes en la Cuantificación Escalar

Finalmente, uniendo las leyes de los procesos de codificación y decodificación, se tiene la ley que gobierna el proceso global de la CE:

$$x \rightarrow Q(x) = c_i \Leftrightarrow x \in (x_i, x_{i+1}] \qquad \textbf{(2.1.2.1.6)}$$

La función característica del cuantificador dibujada en la Figura 2.6 muestra la operación no lineal, irreversible y escalonada que esta cuantificación efectúa sobre las muestras de entrada x.

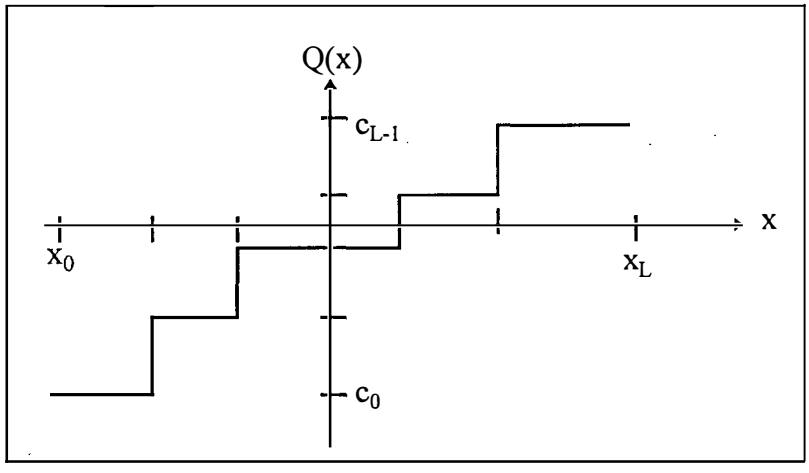

Fig. 2.6 Ley de cuantificación

Utilizando una medida cuadrática del error y teniendo en cuenta el carácter probabilístico de la fuente x, plasmada en su función densidad de probabilidad $f_x(x)$, la distorsión media del proceso viene dada por:

$$D \quad E\{\|x - Q(x)\|_2\} \quad \int_S f_x(x)|x - Q(x)|^2 dx$$

$$= \sum_{i=0}^{L-1} \int_{S_i} f_x(x)|x - Q(x)|^2 dx = \sum_{i=0}^{L-1} \int_{x_i}^{x_{i+1}} f_x(x)|x - c_i|^2 dx \qquad \textbf{(2.1.2.1.7)}$$

En esta expresión aparecen explícitamente los parámetros de diseño del cuantificador:

- la partición (función de decodificación) dada por los puntos x_i

- la librería de códigos dada por los vectores c_i.

2.1.2.2- Cuantificación uniforme

Un cuantificador uniforme es un cuantificador regular en el que los límites x_i están equiespaciados y los valores de salida c_i son los puntos medios de estos intervalos:

$$x_{i+1} - x_i = \Delta \qquad \textbf{(2.1.2.2.1)}$$

$$c_i = \frac{x_i + x_{i+1}}{2} \qquad \textbf{(2.1.2.2.2)}$$

En el caso en que los extremos x_0 y x_L sean finitos, el rango completo de cuantificación será:

$$B=x_L-x_0 \qquad \textbf{(2.1.2.2.3)}$$

Entonces este rango se presenta dividido en L intervalos de cuantificación idénticos y de tamaño

$$\Delta = \frac{B}{L} \qquad \textbf{(2.1.2.2.4)}$$

Asimismo los valores de cuantificación se encontrarán equiespaciados, con una separación de tamaño Δ entre valores consecutivos:

$$c_{i+1} - c_i = \Delta \qquad \textbf{(2.1.2.2.5)}$$

En la figura 2.7 se presenta la función característica de esta cuantificación.

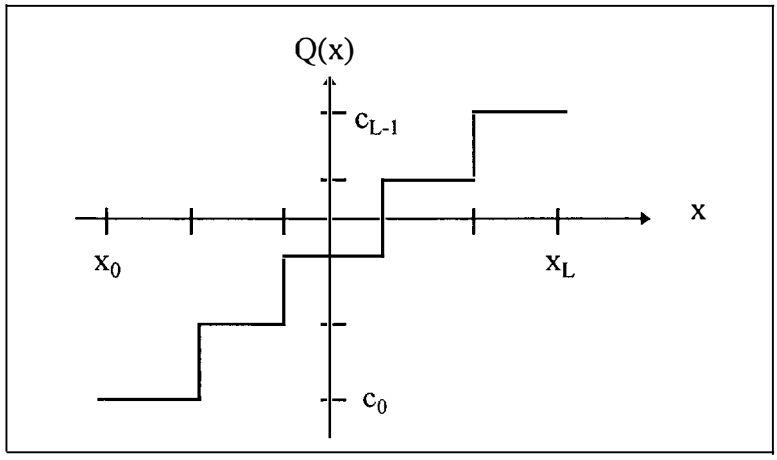

Fig. 2.7 Ley de cuantificación uniforme

Independientemente de la función densidad de probabilidad de x, el máximo error posible es de amplitud $\Delta / 2$. El cuantificador uniforme minimiza este error, lo cal le da una gran robustez y le proporciona un comportamiento razonablemente bueno para gran variedad de señales. Por esta razón y por su simplicidad, es el cauntificador más usado en la conversión A/D.

Cuando la función densidad de probabilidad (f.d.p.) de x está uniformemente distribuida en el rango de variación de x, el error de cuantificación también está uniformemente distribuido en el intervalo $\left(-\frac{\Delta}{2}, \frac{\Delta}{2}\right]$. La distorsión producida en todos los intervalos será igual y de valor cuadrático medio dado por:

$$D_i = \int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} \epsilon^2 \frac{1}{\Delta} d\epsilon = \frac{\Delta^2}{12} \qquad \textbf{(2.1.2.2.6)}$$

que coincidirá con la distorsión media de la cuantificación.

2.1.2.3- Cuantificación no uniforme

Cuando la distribución probabilística de la fuente no es uniforme, el cuantificador uniforme no resulta, en general, óptimo y, en muchos casos, puede originar ruido considerable en la señal cuantificada. La alternativa es el diseño de cuantificadores no uniformes, que sigan leyes de cuantificación apropiadas para las señales con las que van a operar. Además de disminuir el ruido de cuantificación, pueden aumentar el rango dinámico de la entrada sin aumentar el número de niveles, lo cual es especialmente interesante en señales de voz o música, en las que el rango de variación de amplitudes es elevado.

Un modelo general de cuantificación no uniforme, con un número finito de niveles es el mostrado en la Figura 2.8.

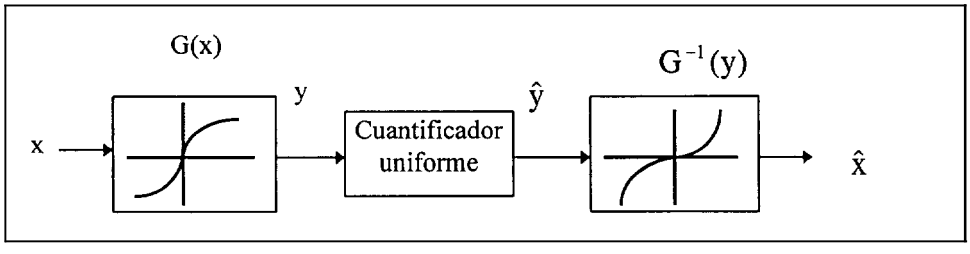

Fig. 2.8 Esquema de cuantificación no uniforme

La entrada x es transformada mediante la función no lineal y monótona G(x) obteniéndose y, que es cuantificada linealmente y sometida al proceso $G^{-1}(\hat{y})$, inverso de G(x).

Al primer módulo se le llama compresor, por el hecho de que normalmente G tiene pendiente suave para valores altos de la entrada y, por tanto, comprime la señal en estos tramos. El tercer módulo, al ser inverso del primero, expande los valores antes comprimidos. El sistema completo recibe el nombre de "compansor" ("compandor", en

terminología anglosajona). En los compansores logarítmicos, la característica de compresión G(x) es simétrica respecto del origen y trata de aproximarse en sus valores positivos a una función logarítmica. Este tipo de cuantificadores ha sido muy utilizado para la compresión de voz y audio, señales en las que la mayor parte de las muestras se encuentran próximas al nivel cero, pero que también contienen una buena porción de amplitudes altas.

El uso de este tipo de cuantificadores permite una elevada Relación Señal a Ruido de Cuantificación en las muestras de pequeño tamaño, mientras que en las de mayor tamaño esta relación no se degrada demasiado, de ahí su profusión y éxito, en particular en los famosos compansores A y μ, cuya ley es lineal a tramos, dentro de una característica con envolvente logarítmica [Gersho 92].

2.1.2.4- Condiciones de optimalidad

El principal objetivo del diseño de un cuantificador es la obtención de unos niveles de cuantificación (libro de códigos) y una regla de codificación (regiones de la partición) que hagan mínima la distorsión dada en (2.1.2.1.7).

En general este problema no tiene una solución cerrada explícita. Sin embargo existen dos condiciones necesarias para la optimalidad, que abren el camino a las diversas soluciones algorítmicas existentes. Ambas condiciones se derivan de la descomposición del cuantificador en los procesos diferenciados de codificación y decodificación.

2.1.2.4.1- Regla del Vecino Más Próximo

Esta regla establece la ley de codificación que minimiza la distorsión para un libro de códigos dado. En esta situación, el decodificador no entra en el diseño, por estar fijado por el libro de códigos y el esquema de la Figura 2.3 puede representarse según la Figura 2.9.

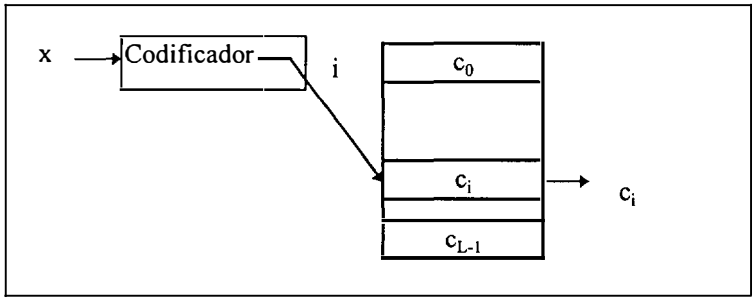

Fig. 2.9 Proceso conjunto de codificación y decodificación en un cuantificador escalar

La ley de codificación óptima en lenguaje natural puede enunciarse como la que "asigna a la muestra de entrada x el índice i correspondiente al nivel c_i que, entre los L posibles niveles del libro de códigos, produzca mínima distorsión". Si para la distorsión se toma una medida de error cuadrática, opción empleada a lo largo de todo el trabajo, esta ley puede formularse de forma analítica como:

$$\mathrm{Cod}(x) = i \Leftrightarrow |x - c_i|^2 \leq |x - c_j|^2 \quad \forall\, j = 0,\ 1,\ \ldots,\ L-1 \qquad \textbf{(2.1.2.4.1.1)}$$

Esto conduce a que las regiones S_i que definen la partición, esto es, aquéllas formadas por todos los puntos x que son codificadores en el índice i, están dadas por:

$$S_i = \left\{ x \in S / |x - c_i|^2 \leq |x - c_j|^2 \right\} \qquad j = 0,\ 1,..,\ L-1 \qquad \textbf{(2.1.2.4.1.2)}$$

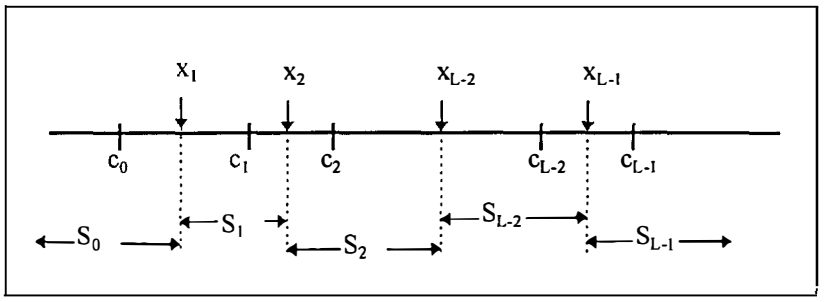

Fig. 2.10 Partición y valores representantes en la Cuantificación Escalar

Volviendo al lenguaje natural, la región S_i está formada por los puntos de S que están más próximos a c_i que a ningún otro c_j del libro de códigos, como puede verse en la Figura 2.10. El nombre de "Vecino Más Próximo" (VMP) queda, pues, sobradamente justificado.

La ley dada en (2.1.2.4.1.1) o en (2.1.2.4.1.2) implica que los puntos x_i que delimitan las regiones contiguas S_{i-1} y S_i se encuentran equidistantes de los niveles c_{i-1} y c_i, según se puede observar también en la anterior figura.

La Regla del VMP, cuya formulación persigue obtener mínima distorsión de cuantificación, verdaderamente consigue minimizar la distorsión media del proceso. En efecto, teniendo en cuenta (2.1.2.1.7), se tiene que:

$$D = \int_S f_x(x)|x - c_i|^2 dx \leq \int_S f_x(x)|x - c_j|^2 dx \qquad \textbf{(2.1.2.4.1.3)}$$

donde la primera integral se refiere a la distorsión media con la regla de codificación dada por el VMP:

$$Cod(x)=i, \quad Decod(i)=c_i \qquad \textbf{(2.1.2.4.1.4)}$$

y la segunda integral se refiere a la distorsión media dada cualquier otra regla de codificación:

$$Cod(x)=j \qquad Decod(j)=c_j \qquad \textbf{(2.1.2.4.1.5)}$$

2.1.2.4.2- Regla de los Centroides

La segunda condición necesaria para la optimalidad se obtiene considerando fija la regla de codificación (partición) y hallando los niveles de cuantificación (libro de códigos) que minimizan la distorsión media del proceso dada por:

$$D(c_i) = \sum_{i\ 0}^{L-1} \int_{S_i} f_x(x)|x - c_i|^2 dx \qquad \textbf{(2.1.2.4.2.1)}$$

La minimización de cada nivel c_i en cada región S_i puede realizarse de manera separada minimizando las funciones

$$D_i = \int_{S_i} f_x(x)|x - c_i|^2 dx \ , \qquad i=0,1, \ , L\text{-}1 \qquad \textbf{(2.1.2.4.2.2)}$$

Utilizando el Cálculo Diferencial convencional, se llega a que los niveles c_i buscados están definidos por [Gersho 92]:

$$c_i = \frac{\int_{S_i} f_x(x)x\, dx}{\int_{S_i} f_x(x)\, dx} = \int_{S_i} f_{x/S_i}(x)\, dx = E\{x / x \in S_i\} \qquad \textbf{(2.1.2.4.2.3)}$$

donde $f_{x/S_i}(x)$ denota la función densidad de probabilidad de la muestra x condicionada a encontrarse en la región S_i y $E\{x / x \in S_i\}$ es la esperanza de x dentro de esta región.

Puede observarse que los resultantes valores c_i coinciden con los valores promedio (centros de masa) de las regiones S_i.

Estas dos condiciones de optimalidad fueron publicadas por primera vez por Lukaszewicz y Steinhauss en 1955 [Lukaszewicz 55] y observadas por Lloyd [Lloyd 57] y más tarde por Max [Max 60], quienes las aplicaron al diseño de cuantificadores escalares. Es por ello que los cuantificadores que satisfacen ambas condiciones reciben el nombre de cuantificadores Lloyd-Max.

2.1.2.5- El algoritmo de Lloyd

En general las dos condiciones de optimalidad no son suficientes para garantizar que el cuantificador sea óptimo. El mismo Lloyd mostró contraejemplos de esta circunstancia [Lloyd 57]. Sin embargo, se pueden formular algoritmos basados en ellas que den soluciones localmente óptimas.

Se presenta a continuación el procedimiento formulado por Lloyd en la anterior referencia. Su algoritmo parte de la generación de un libro de códigos inicial. Aplicando la Regla del VMP se obtiene la partición óptima para este libro, tras la cual se aplica la Regla de los Centroides, que obtiene la librería óptima para la recién calculada partición. Estas dos reglas se van aplicando sucesivas veces de forma alternada, hasta que se decide acabar el proceso, generalmente cuando éste deja de progresar de manera apreciable.

Llamando C[m], S[m] y D[m] al libro de códigos, la partición y la distorsión en la iteración m-ésima respectivamente, definimos la iteración el Lloyd como la aplicación de la Regla del VMP seguido de la de los Centroides (Figura 2.11). El algoritmo de Lloyd queda expresado en los diagramas de la Figura 2.12.

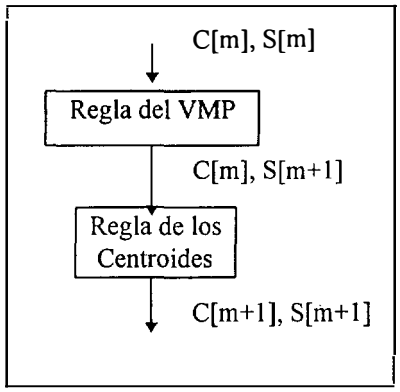

Fig. 2.11. Iteración de Lloyd

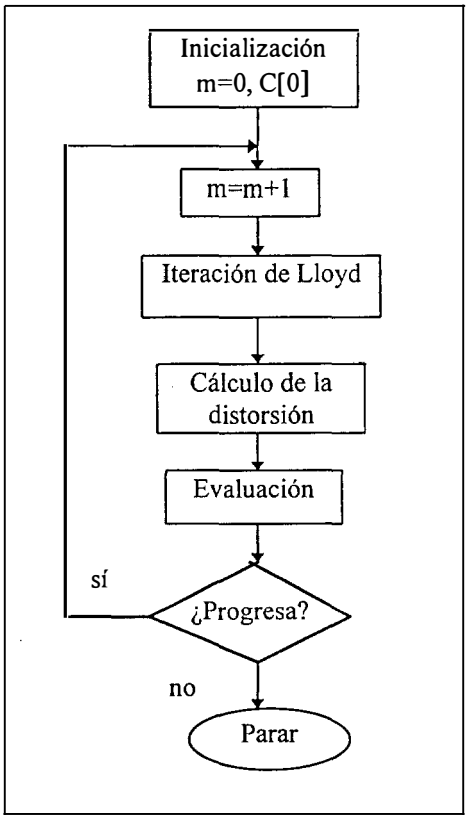

Fig. 2.12.a Algoritmo de Lloyd (diagrama de flujo)

Paso 1. Inicialización del libro de códigos
Paso 2. Aplicación de la Regla del VMP
Paso 3. Aplicación de la Regla de los Centroides
Paso 4. SI la distorsión se reduce, saltar al Paso 2
Paso 5. SI NO, parar

Fig. 2.12.b Algoritmo de Lloyd (pseudocódigo)

Las dos reglas de optimalidad aseguran la evolución siempre decreciente de la distorsión. El criterio de parada que normalmente se utiliza es el de observar si la disminución relativa de la distorsión

$$D_r[m] \quad \frac{D[m-1]-D[m]}{D[m]} \qquad \textbf{(2.1.2.5.1)}$$

supera un cierto umbral. Si es así, se continúan las iteraciones, si no, se finaliza el proceso.

2.1.2.6- Diseño basado en datos empíricos

En la práctica es difícil encontrar modelos analíticos precisos que describan las distribuciones probabilísticas de las fuentes de señal que se quieren cuantificar. Lo más habitual es contar con un número suficientemente grande de muestras, que alberguen una descripción aproximada de la distribución estadística de estas señales.

Llamemos "conjunto de entrenamiento" a este conjunto muestral:

$$V = \{v_0, v_1, \ , \ v_{N-1}\} \qquad \textbf{(2.1.2.6.1)}$$

La teoría estadística nos permite asegurar [Papantoni 79] que la probabilidad relativa de encontrar algún elemento de V en el intervalo $[x, x+\Delta]$, definida según:

$$\mathrm{Prob}(x \in [x, x+\Delta]) = \frac{N_\Delta}{N} \qquad \textbf{(2.1.2.6.2)}$$

(N_Δ es el número de estos elementos contenidos en el anterior intervalo y N es el número total de elementos de V) se aproxima a la función densidad de probabilidad de x, a medida que N crece y Δ disminuye:

$$\lim_{\substack{N\to\infty \\ \Delta\to 0}} \{\mathrm{Prob}(x \in [x, x+\Delta])\} = f_X(x) \qquad \textbf{(2.1.2.6.3)}$$

En estos casos, un camino válido para aplicar la Regla de los Centroides sería inferir, a partir de V, un modelo analítico de la f.d.p. de la fuente x, a partir del cual llevar a cabo las integraciones de esta regla, dadas en (2.1.2.4.2.3).

Una segunda opción, mucho más sencilla computacionalmente, ya que evita las no pocas complicaciones de la obtención de la $f_x(x)$ y de la integración numérica mencionada, es la de reformular la Regla de los Centroides considerando que el espacio S de la posibles entradas ahora es sustituido por V, esto es, por un espacio discreto de puntos conocidos.

La función de distorsión será ahora:

$$D = \sum_{i=0}^{L-1} \sum_{v_{ij} \in V_i} |v_{ij} - c_i|^2 = \sum_{i=0}^{L-1} \sum_{j=0}^{N_i-1} |v_{ij} - c_i|^2 \qquad \textbf{(2.1.2.6.4)}$$

donde V_i constituye la subpartición i del espacio V formada por las muestras de V que son codificadas con el índice i:

$$V_i \quad \{v \in V / \mathrm{Cod}(v) \quad i\} = \{v_{ij}, \ j \quad 0, 1, .., N_i \quad 1\} \qquad \textbf{(2.1.2.6.5)}$$

La minimización de la función de distorsión D dada en (2.1.2.6.4) con respecto a c_i, conduce a la versión discreta y determinística de la Regla de los Centroides [Gersho 92]:

$$c_i = \frac{1}{N_i} \sum_{v_{ij} \in V_i} v_{ij} \qquad \textbf{(2.1.2.6.6)}$$

Puede notarse que nuevamente los puntos c_i corresponden a los centros de masa de las subparticiones V_i.

A su vez, la Regla del VMP, atendiendo exactamente al mismo razonamiento que se veía en el diseño a partir de funciones probabilísticas, puede expresarse según:

$$\text{Cod}(v) = i \Leftrightarrow \left|v \quad c_i\right|^2 \leq \left|v \quad c_j\right|^2 \quad \forall\, j = 0,\ 2,\ ..,\ L-1 \qquad \textbf{(2.1.2.6.7)}$$

lo cual conduce a que las subparticiones óptimas de V sean ahora:

$$V_i = \left\{ v \in V \,/\, \left|v - c_i\right|^2 \leq \left|v - c_j\right|^2 \quad \forall\, j = 0,\ 2,\ ..,\ L-1 \right\} \qquad \textbf{(2.1.2.6.8)}$$

El Algoritmo de Lloyd descrito en la Figura 2.14 podrá ser aplicado, únicamente modificando las reglas anteriores y utilizando la función de distorsión dada en (2.1.2.6.4).

2.1.3- Cuantificación Vectorial sin errores de canal

En este apartado se extiende el concepto de cuantificación a variables de varias dimensiones. Unicamente se tratará aquí el caso en el que el canal no introduce errores, dejándose para secciones siguientes el caso del canal ruidoso.

La consideración de ausencia de errores en el canal tiene una motivación diversa. Por una parte ésta ha sido una premisa en el diseño de cuantificadores vectoriales en los que, debido a la tecnología empleada o a la protección adicional incorporada, los errores de transmisión se consideraban despreciables.

Por otra parte, esta formulación constituye un punto de partida más sencillo a la hora de diseñar cuantificadores vectoriales que sí tienen presente los errores en el canal. Partiendo de cuantificadores concebidos para operar sin ruido, bien añadiendo bits redundantes para evitar o corregir posibles errores de canal [Atal 93], o bien reordenando las asignaciones de vectores a símbolos transmitidos para minimizar el efecto de tales errores [Wu 93], [Zeger 87], se han conseguido soluciones de buenas prestaciones en este tipo de problemas.

Por último, buena parte de las técnicas y algoritmos de diseño de CV proceden de una natural extensión al caso del canal ruidoso de sus versiones ideadas para condiciones de canal sin ruido (véanse [Farvardin 87] y [Farvardin 91] por una parte, y [Linde 80] y [Kumazawa 84] por otra, por citar dos ejemplos importantes).

2.1.3.1- Planteamiento

La CV se plantea en los siguientes términos. Sea S el espacio de vectores que se quiere cuantificar, que se supone conexo y de dimensión D. Sea $\bar{x}$ un vector genérico perteneciente a S. Se pretende representar este espacio mediante el conjunto de L vectores, también de dimensión J:

$$\{C\} = \{\bar{c}_0, \bar{c}_1, \ .., \bar{c}_{L-1}\} \qquad \textbf{(2.1.3.1.1)}$$

que recibe el nombre de libro o librería de códigos. A su vez, los vectores de $\{C\}$ son llamados vectores código.

Al igual que en el caso escalar, se puede dividir la CV en dos procesos diferenciados, el de codificación y el de decodificación (Figura 2.13).

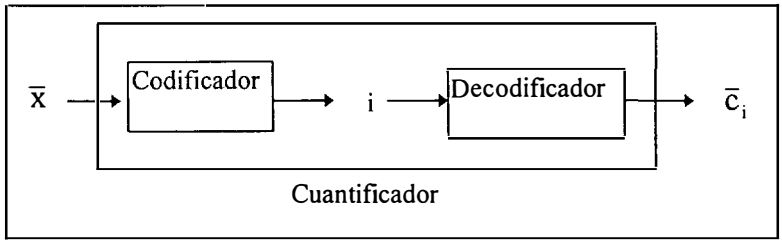

Fig. 2.13 Diagrama funcional de un cuantificador vectorial

Con la codificación se asigna al vector $\bar{x}$ el índice i perteneciente al intervalo $\{0, 1, .., L-1\}$, siguiendo una ley de codificación concretada "a priori".

También ahora la regla de codificación impone una partición del espacio S en regiones o subparticiones S_i dadas por:

$$S_i = \{\bar{x} \in S / Cod(\bar{x}) = i\}, \quad i=0, 1, .. L-1 \qquad \textbf{(2.1.3.1.2)}$$

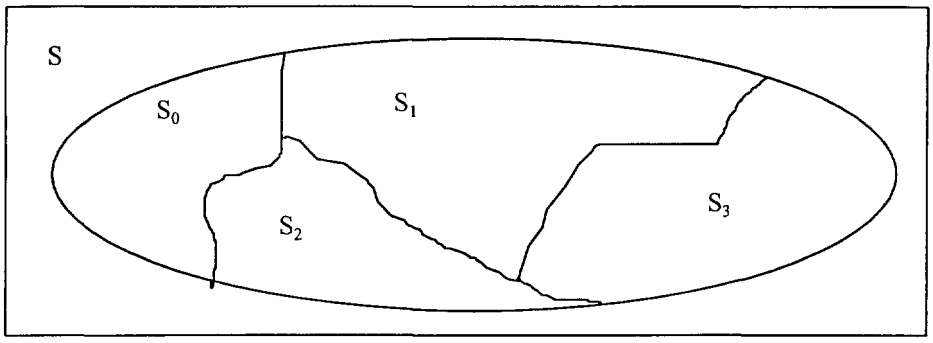

Fig. 2.14 Partición del espacio S en los subespacios S_i (i=1,2,3,4).

Tal como aparece en la Figura 2.14, los subespacios S_i forman una partición del espacio S:

$$\bigcup_{i=0}^{L-1} S_i = S \qquad \textbf{(2.1.3.1.3.a)}$$

$$\bigcap_{i=0}^{L-1} S_i = \varnothing \qquad \textbf{(2.1.3.1.3.b)}$$

Dicha partición define, a su vez y sin ambigüedad, la regla de codificación.
En la decodificación se extrae el vector código $\bar{c}_i$ al recibirse el índice i:

$$Decod(i) = \bar{c}_i \qquad \textbf{(2.1.3.1.4)}$$

Globalmente, mediante la cuantificación, el vector $\bar{x}$ es convertido en uno de los vectores código, en particular en aquél asociado a la región S_i que contiene a $\bar{x}$ (Figura 2.15):

$$\bar{x} \rightarrow Q(\bar{x}) = \bar{c}_i | x \in S_i \qquad \textbf{(2.1.3.1.5)}$$

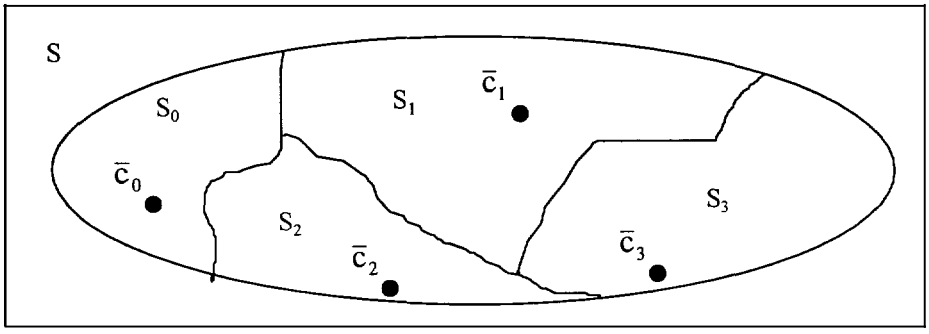

Fig. 2.15 Vectores código $\bar{c}_i$ asociados a las regiones S_i.

Cuanto más se aproxime la salida $\bar{c}_i$ a la entrada $\bar{x}$, mejor se considerará el cuantificador. Se llama error de cuantificación a la diferencia entre los valores de entrada y salida:

$$\bar{\varepsilon} = \bar{x} - \bar{c}_i \qquad \textbf{(2.1.3.1.6)}$$

Como, en general, $\bar{x}$ es un vector genérico de S que atiende a una distribución probabilística (conocida o no), la figura de ruido que se tiene en cuanta es el error cuadrático medio de la cuantificación, o distorsión cuadrática media, dada por la esperanza del módulo al cuadrado del vector $\bar{\varepsilon}$:

$$D = E\left\{\left\|\bar{x} - \bar{c}_i\right\|_2\right\} = E\left\{\left\|\bar{x} - Q(\bar{x})\right\|_2\right\} \qquad \textbf{(2.1.3.1.7)}$$

donde el símbolo $E(v)$ define la esperanza de una variable aleatoria genérica v, y $\left\|\bar{a} - \bar{b}\right\|_2$ es la distancia euclídea entre dos vectores genéricos $\bar{a}$ y $\bar{b}$.

Esta distorsión es función de la distribución probabilística de la señal $\bar{x}$, a través de su f.d.p. f_x, del libro de códigos C y de la regla de decisión del decodificador dada por la partición de S en las regiones S_i. Con todo ello, la distorsión anterior puede obtenerse por medio de la expresión integral:

$$D = \sum_{i=0}^{L-1} \int_{S_i} f_x(\bar{x}) \left\|\bar{x} - Q(\bar{x})\right\|_2 d\bar{x} \qquad \textbf{(2.1.3.1.8)}$$

El discurso de los párrafos anteriores se sitúa sobre la panorámica de la compresión con el fin de obtener representaciones compactas de la señal de cara a su almacenamiento. Si cambiamos al plano de la comunicación de señales, los bloques de codificación y decodificación anteriores siguen siendo válidos, pero ahora el índice i va a ser enviado por un canal de transmisión. Si éste no introduce errores, el mismo índice i será obtenido en recepción, al otro extremo del canal, por lo que la Figura 2.1 seguirá sirviendo para describir el sistema, y todas las explicaciones y expresiones anteriores continuarán siendo válidas. Si, por el contrario, el canal es ruidoso y genera errores en la transmisión, no se tendrá la certeza de que al final de ésta llegue el mismo índice i que se envió y la CV precisará una pequeña reformulación. Esto se detallará en el apartado 2.1.4.

2.1.3.2- Diseño de cuantificadores óptimos

La distorsión anterior es función de la regla de codificación y del libro de códigos. Al igual que en el caso escalar, no existe una solución explícita para minimizar esta función. Sin embargo, pueden encontrarse dos condiciones necesarias de optimalidad, extensiones al caso vectorial de las Reglas VMP y de los Centroides.

2.1.3.2.1- Regla del Vecino Más Próximo

Partiendo de un libro de códigos $\{C\}$ fijo, esta ley establece cuál debe ser la regla de codificación (partición del espacio S) que hace mínima la distorsión D. El mismo razonamiento aplicado en la sección 2.1.2.4.1 a la CE conduce, en el caso vectorial, a la regla de decisión dada por:

$$Cod(\bar{x}) = i \Leftrightarrow \left\|\bar{x} - \bar{c}_i\right\|_2 \leq \left\|\bar{x} - \bar{c}_j\right\|_2 \quad \forall\, j = 0,\ 1,\ ..,\ L-1 \qquad \textbf{(2.1.3.2.1.1)}$$

que asigna al vector de entrada $\bar{x}$ el índice que identifica al vector código más próximo a él (que produce error cuadrático mínimo) entre todos los vectores código. Esta ley conduce a una partición del espacio S en regiones S_i dadas por:

$$S_i = \left\{ \bar{x} \in S / \left\| \bar{x} - \bar{c}_i \right\|_2 \leq \left\| \bar{x} - \bar{c}_j \right\|_2 \right\} \qquad j = 0,\ 1,..,\ L-1 \qquad \textbf{(2.1.3.2.1.2)}$$

2.1.3.2.2- Regla de los Centroides

Considerando fija la regla de codificación (partición del espacio S), el mejor conjunto de vectores código será aquél que minimice la distorsión dada en (2.1.3.1.8).

Aplicando de nuevo elementos básicos de Cálculo, puede llegarse a que estos vectores están dados por [Gersho 92]:

$$\bar{c}_i = \frac{\int_{S_i} f_x(\bar{x})\bar{x}\, d\bar{x}}{\int_{S_i} f_x(\bar{x})\, d\bar{x}} = \int_{S_i} f_{x/S_i}(\bar{x})\, d\bar{x} = E\left\{ \bar{x} / \bar{x} \in S_i \right\} \qquad \textbf{(2.1.3.2.2.1)}$$

donde $f_{x/S_i}(\bar{x})$ denota la función densidad de probabilidad del vector $\bar{x}$ condicionado a encontrarse en la región S_i y $E\left\{ \bar{x} / \bar{x} \in S_i \right\}$ es la esperanza de $\bar{x}$ dentro de esta región.

Puede observarse que estos vectores coinciden con los centros de masa (centroides) de cada una de las regiones S_i, de ahí el nombre de esta regla.

2.1.3.3- El algoritmo LBG

Al igual que en el caso escalar, las dos reglas vistas son necesarias pero no suficientes para lograr la optimalidad. A continuación se presenta un algoritmo en el que se van sucediendo alternada e iterativamente ambas reglas. Este algoritmo no es sino la extensión natural del algoritmo de Lloyd considerando que ahora, en el caso vectorial, se han de aplicar las correspondientes versiones de las Reglas del VMP y de los Centroides, así como la nueva función de distorsión dada en (2.1.3.1.8). Por ello, es frecuentemente referenciado como algoritmo de Lloyd Generalizado o GLA (Generalized Lloyd Algorithm). También se le da el nombre de algoritmo LBG por ser los autores Y. Linde, A. Buzo y R. M. Gray quienes introdujeron primero esta generalización en el diseño de cuantificadores vectoriales [Linde 80], aunque la idea ya había sido apuntada antes en el contexto de la clasificación estadística [Forgey 65], [MacQueen 67].

Aquí se adopta el segundo acrónimo, dejando el término GLA para la generalización que surge al considerar posibles errores en la transmisión debidos al ruido en el canal, tema que será tratado en apartados posteriores.

En definitiva, el algoritmo LBG puede ilustrarse con idénticos diagramas funcionales y pseudocódigo que el algoritmo de Lloyd.

Llamando ahora iteración LBG al bucle básico basado en aplicar la nuevas Reglas del VMP y de los Centroides (Figura 2.16), se puede representar el funcionamiento del

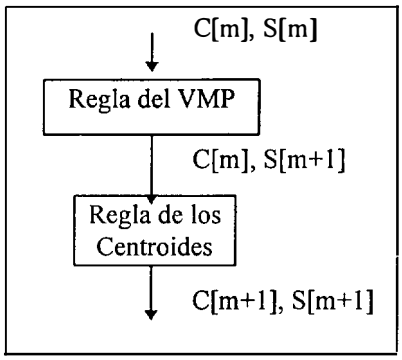

Fig. 2.16 Iteración LBG

algoritmo completo, según los diagramas de la Figura 2.17.

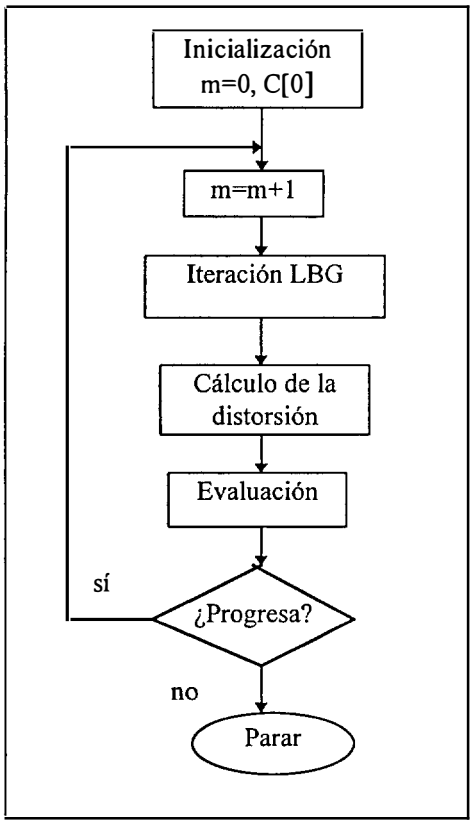

Fig. 2.17.a Algoritmo LBG (diagrama de flujo)

Paso 1. Inicialización del libro de códigos
Paso 2. Aplicación de la Regla del VMP
Paso 3. Aplicación de la Regla de los Centroides
Paso 4. SI la distorsión se reduce, saltar al Paso 2
Paso 5. SI NO, parar

Fig. 2.17.b Algoritmo LBG (pseudocódigo)

Igualmente el algoritmo se detendrá cuando el descenso de la distorsión relativa

$$D_r[m] = \frac{D[m-1] - D[m]}{D[m]} \qquad \textbf{(2.1.3.3.1)}$$

en una iteración del bucle no supere un umbral mínimo.

2.1.3.4- Diseño basado en datos empíricos

Nuevamente, como la descripción analítica de la f.d.p. del vector $\bar{x}$ no suele ser frecuente, se recurre a utilizar un conjunto de entrenamiento

$$V = \{\bar{v}_0, \bar{v}_1, .., \bar{v}_{N-1}\} \qquad \textbf{(2.1.3.4.1)}$$

que represente estadísticamente la distribución de $\bar{x}$.

Paralelos a los desarrollos del caso escalar encontramos las nuevas formulaciones para la distorsión media del proceso y las Reglas del VMP y de los Centroides.

La distorsión media quedará:

$$D = \sum_{p=0}^{N-1} \left\| \bar{v}_p - Cod(\bar{v}_p) \right\|_2 \qquad \textbf{(2.1.3.4.2)}$$

Llamando V_i al subconjunto de V con aquellos vectores que son codificados con el índice i, esto es:

$$V_i = \{\bar{v} \in V / Cod(\bar{v}) = i\} \qquad \textbf{(2.1.3.4.3)}$$

se tendrá que la distorsión media dada en (2.1.3.4.2) podrá ponerse como la suma de las distorsiones parciales D_i asociadas a cada región [Gersho 92]:

$$D = \sum_{i=0}^{L-1} D_i = \sum_{i=0}^{L-1} \sum_{j=0}^{N_i-1} \left\| \bar{v}_{ij} - \bar{c}_i \right\|_2 \qquad \textbf{(2.1.3.4.4)}$$

La Regla del Vecino Más Próximo se formulará ahora según:

$$Cod(\bar{v}) = i \Leftrightarrow \left\| \bar{v} - \bar{c}_i \right\|^2 \le \left\| \bar{v} - \bar{c}_j \right\|^2 \quad \forall\, j = 0, 1, .., L-1 \qquad \textbf{(2.1.3.4.5)}$$

lo cual implica que las partición del conjunto de entrenamiento V esté formada por los subconjuntos V_i dados por:

$$V_i = \left\{ \bar{v} \in S / \left\| \bar{v} - \bar{c}_i \right\|^2 \le \left\| \bar{v} - \bar{c}_j \right\|^2 \right\} \qquad j = 0, 1, .., L-1 \qquad \textbf{(2.1.3.4.6)}$$

La Regla de los Centroides será ahora [Gersho 92]:

$$\bar{c}_i = \frac{1}{N_i} \sum_{j=0}^{N_i-1} \bar{v}_{ij} \qquad i=0, 1, L-1 \qquad \textbf{(2.1.3.4.7)}$$

Conviene destacar el caso en que alguna de estas regiones se encuentre vacía. Si esto es así esta última expresión no sería válida para el índice i correspondiente con esa región. Normalmente este problema se arregla eliminando dicha región y dividiendo alguna región no vacía en dos. Puede tomarse para ello la región con mayor distorsión D_i, aquélla con mayor distorsión relativa D_i/N_i o bien la región más poblada.

Para la división de estas regiones [Furui 89] se toman dos vectores código muy próximos al vector código de la región elegida y se aplica sobre ellos la Regla del VMP.

El cuanto al algoritmo LBG, éste podrá aplicarse en el caso empírico tal y como se describe en el apartado 2.1.3.3 teniendo en cuenta la nueva formulación de las Reglas de optimalidad y de la distorsión.

2.1.3.5- Cuantificación Vectorial Constreñida

La CV tal como se ha descrito en los apartados anteriores es la manera más eficiente de representar un vector cuya distribución de probabilidades se conoce, bien formalmente a través de su f.d.p., bien estimativamente, mediante un conjunto de vectores de entrenamiento suficientemente representativo. Sin embargo, cuando la librería de códigos es grande, tanto la complejidad en la búsqueda del VMP en el codificador como los requerimientos de almacenamiento en el codificador y en el decodificador, ambos proporcionales al tamaño de esta librería, pueden llegar a ser excesivos. Por otra parte, en la fase de diseño es conveniente utilizar conjuntos de prueba de varios miles de muestras (decenas o incluso centenares de miles de píxeles, en el caso de imágenes).

Todo ello impone severas limitaciones en el tamaño de las librerías, lo cual redunda, a su vez, en unas prestaciones quizás menos elevadas de lo deseable.

Es corriente reducir la complejidad de los cuantificadores vectoriales imponiendo restricciones a su manera de operar, de forma que la búsqueda se realice sobre conjuntos más reducidos de vectores. Ello conducirá a representaciones no óptimas del espacio de muestras de entrada, pero hará posible utilizar libros de códigos más extensos, con los que se podrán alcanzar más altas prestaciones que en el caso general sin restricciones.

En la práctica estos sistemas son mucho más comunes que la CV no sujeta a restricción, existiendo una inmensa variedad de tipos, y versiones híbridas, no dejando de aparecer en la bibliografía especializada. Entre ellos cabe destacar los cuantificadores estructurados en árbol, clasificados, transformados, particionados, sin media, ganancia-forma, multietapa, jerárquicos, etc.

2.1.3.6- Cuantificación Vectorial con memoria

La CV detallada en las anteriores secciones y las enunciadas en el apartado precedente tratan de eliminar la redundancia entre las muestras que componen el vector. Esto es, el carácter de estas codificaciones es vector a vector, sin tener presente la posible correlación entre vectores próximos en el tiempo o adyacentes en el espacio. Si verdaderamente esta correlación es significativa, resulta interesante poder hacer uso de ella en el proceso de cuantificación. Entramos en el terreno de la CV con memoria. En esta línea, los cuantificadores vectoriales predictivos tratan de predecir un vector de muestras a partir de vectores anteriores; seguidamente extraen la diferencia entre el vector de entradas y su predicción y cuantifican vectorialmente este vector diferencia.

Por otra parte, cuando las características estadísticas de la entrada varían en el tiempo (o en las distintas dimensiones espaciales, si nos referimos a imágenes) un CV fijo, como los vistos hasta ahora presenta graves inconvenientes. Para mantener una calidad aceptable debe ser capaz de cubrir todo el rango de variación del vector de entradas, lo cual supone un tamaño muy grande para la librería de códigos. Con libros

de códigos moderados los cuantificadores vectoriales presentarán distorsiones elevadas. Los cuantificadores vectoriales adaptativos son aquéllos en los que la librería de códigos varía a lo largo del tiempo (o de las variables independientes existentes) para adecuarse a la estadística local observada a la entrada. Por otro lado hay cuantificadores que unen estos conceptos con los esquemas de la cuantificación constreñida, de forma que existen cuantificadores en los que se van adaptando las medias, o las ganancias, etc.

2.1.3.7- Cuantificación Vectorial de tasa de bits variable

La asignación de diferente número de bits a los distintos vectores cuantificados proporciona un nuevo grado de libertad con el que componer codificadores más eficientes, llamados "de tasa de bits variable".

Algunas aplicaciones de compresión no están diseñadas para esperar una cadencia regular de muestras a su entrada, en particular aquéllas que tienen que ver con almacenamiento de audio o vídeo o con redes de comunicación de paquetes. Son por ello especialmente adecuadas para el empleo de estas técnicas de codificación. Tal es el caso de la compresión y recuperación de bases de datos de imágenes o la distribución de vídeo bajo demanda.

Si, por el contrario, el sistema posee una tasa de transmisión fija, la utilización de codificación de tasa de bits variable requiere la inclusión de hardware adicional para el almacenamiento y sincronización de los datos. Incluso en estos casos puede ser interesante su uso si la calidad final del sistema así lo indica.

Incorporando diversas estructuras de la CV constreñida, se formulan distintas maneras de construir cuantificadores de tasa de bits variable: CV de dimensión variable, CV forma-ganacia de tasa variable, CV multietapa de tasa variable, CV estructurado en árbol podado, etc.

Tanto las técnicas de VQ constreñido, como las diferenciales o adaptativas, abarcadas en la categoría de VQ con memoria, o las codificaciones de tasa de bits variable se encuentra fuera del planteamiento y objetivo de la presente Tesis, por lo que no se describen con más detalle. Muchas de ellos junto con otras técnicas asociadas a la CV se recogen en el libro de A. Gersho y R. M. Gray [Gersho 92], obra verdaderamente indispensable en cualquier estudio sobre esta materia.

2.1.4- Cuantificación Vectorial con errores de canal

Las aplicaciones de telefonía móvil y comunicaciones por satélite, de tanta expansión en nuestra década hacen imprescindible la consideración de los posibles errores en la transmisión de las señales a través de estos sistemas [Atal 93], [Gibson 96]. En este apartado se revisan y generalizan a la nueva situación de canal ruidoso los conceptos, planteamientos y técnicas de diseño vistos en el apartado anterior.

2.1.4.1- Planteamiento

En presencia de ruido en el canal el esquema de la CV es el de la Fig. 2.18.

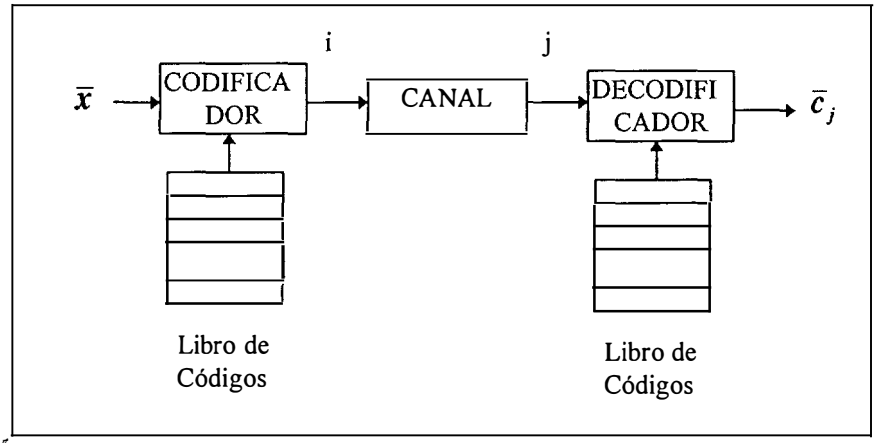

Figura 2.18 Sistema de Cuantificación Vectorial en presencia de errores de canal

La distorsión debida a la disparidad entre la entrada $\bar{x}$ y la salida $\bar{c}_j$ resulta de la combinación de dos factores:

- el propio proceso de cuantificación,
- el posible error en la transmisión del índice i.

Para analizar con mayor facilidad esta segunda componente es conveniente describir la CV con un poco más de detalle. En vez de suponer que i es enviado directamente, asumiremos que existe un alfabeto de L símbolos $B = \{b_0, b_1, \ , b_{L-1}\}$ binarios o de otro tipo, aptos para la transmisión.

En un principio puede pensarse ("asignación natural") que el índice i es transmitido a través del símbolo b_i, y que en recepción, si llega el símbolo b_j, se asociará al índice j.

En caso de no existir errores de canal, qué asociación se establezca entre índices y símbolos es irrelevante, pues la distorsión del proceso sólo tendrá que ver con el proceso de cuantificación, esto es, con la librería de códigos y con la regla de codificación. Sin embargo, cuando el canal es ruidoso y se originan errores en la transmisión, tal asociación puede jugar un papel importante en las prestaciones del sistema. Según la Figura 2.19, debido al ruido de canal, se recibe el símbolo b_j, aunque el transmitido había sido el b_i, por lo que se obtiene a la salida el vector código $\bar{c}_j$ que dista de la entrada $\bar{x}$ una cantidad dada por $\|\bar{x} - \bar{c}_j\|_2$.

Si en vez del conjunto B, se hubiera tenido el conjunto B', formado por los mismos elementos que B, pero ordenados de diferente manera, tras enviar el símbolo b_i', se habría recogido en recepción el símbolo b_j'. El vector a la salida habría sido el

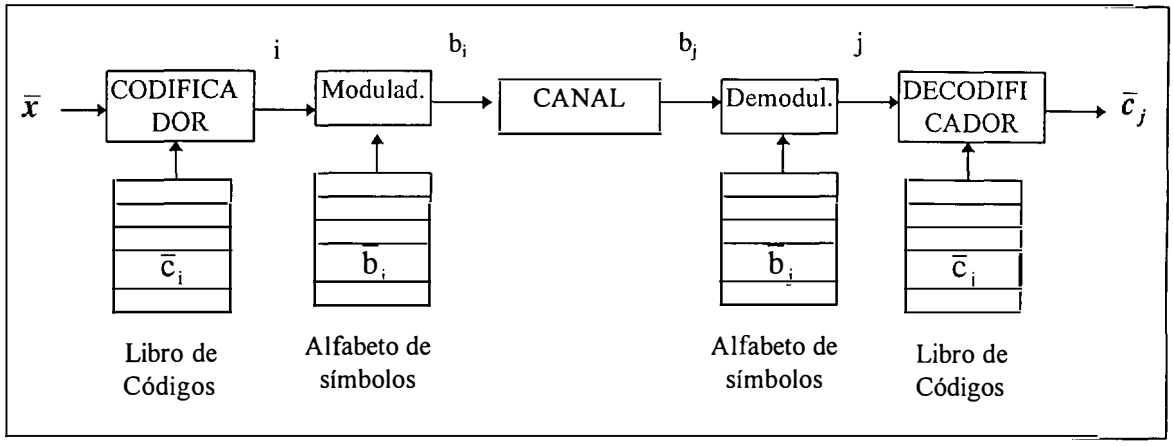

Figura 2.19 Sistema de Cuantificación Vectorial en presencia de errores de canal

$\bar{c}_j$', que distaría de la entrada la cantidad $\|\bar{x} - \bar{c}_j'\|_2$.

En general,

$$\|\bar{x} - \bar{c}_j\|_2 \neq \|\bar{x} - \bar{c}_j'\|_2 \qquad \textbf{(2.1.4.1.1)}$$

por lo que, dado un mismo conjunto de símbolos, dependiendo de la asignación que se haga entre éstos y los índices resultantes del codificador, se obtendrán mejores o peores cuantificadores vectoriales.

En este punto es conveniente introducir la función Π que establece una asignación biunívoca entre los L índices del libro de códigos C y los del alfabeto B[(*)], basado en la siguiente directriz: "cuando el índice i sea seleccionado en el codificador, se transmitirá el símbolo $b_{i'}$, elemento i'-ésimo de B".

Formalmente:

$$i \rightarrow \Pi(i) = i' \qquad \begin{matrix} i = 0,\ 1,\ ..,\ L-1 \\ i' = 0,\ 1,\ ..,\ L-1 \end{matrix} \qquad \textbf{(2.1.4.1.2)}$$

$$i' \rightarrow B(i') = b_{i'} \qquad \textbf{(2.1.4.1.3)}$$

(*) En último término, puede entenderse que la función de asignación Π es la que define la asociación biunívoca entre los índices del libro de códigos C y los símbolos del alfabeto B. Sin embargo, con el ánimo de clarificar las explicaciones se presenta esta correspondencia en dos pasos separados: asignación de índices y decodificación de símbolos.

Este mecanismo constituye el bloque que hemos llamado "modulador" y aparece desglosado en la Figura 2.20.

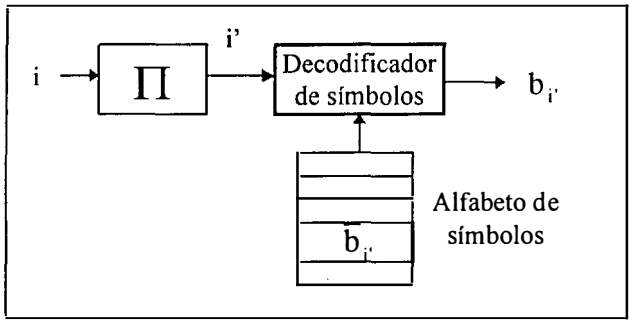

Fig. 2.20 Esquema del "modulador"

La función de asignación Π puede expresarse fácilmente por medio de la lista

$$B_{\Pi} = \{\Pi(0), \Pi(1), \quad , \Pi(L-1)\} \qquad \textbf{(2.1.4.1.4)}$$

que será una de las L! permutaciones posibles de la lista

$$B_{\Pi_N} = \{0, 1, \quad , L-1\} \qquad \textbf{(2.1.4.1.5)}$$

correspondiente a la "asignación natural".

El bloque denominado "demodulador" tiene función y estructura inversa al modulador (Figura 2.21).

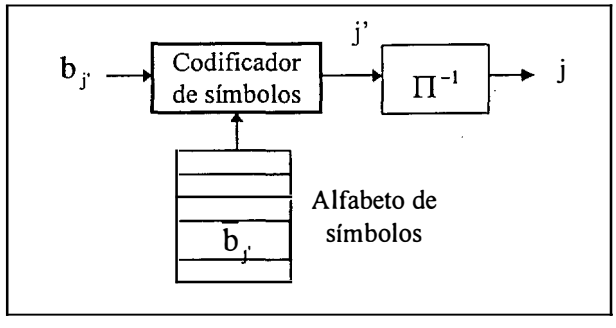

Fig. 2.21 Esquema del "demodulador"

El codificador de símbolos extrae el índice j' cuando el llega el símbolo $b_{j'}$:

$$B^{-1}\left(b_{j'}\right) = j' \qquad \textbf{(2.1.4.1.6)}$$

A continuación la función Π^{-1}, inversa a la función de asignación Π, transforma el índice j' en el j, que será entregado al decodificador:

$$\Pi(j') = j \qquad \textbf{(2.1.4.1.7)}$$

Globalmente, se tendrá el esquema de la Figura 2.22.

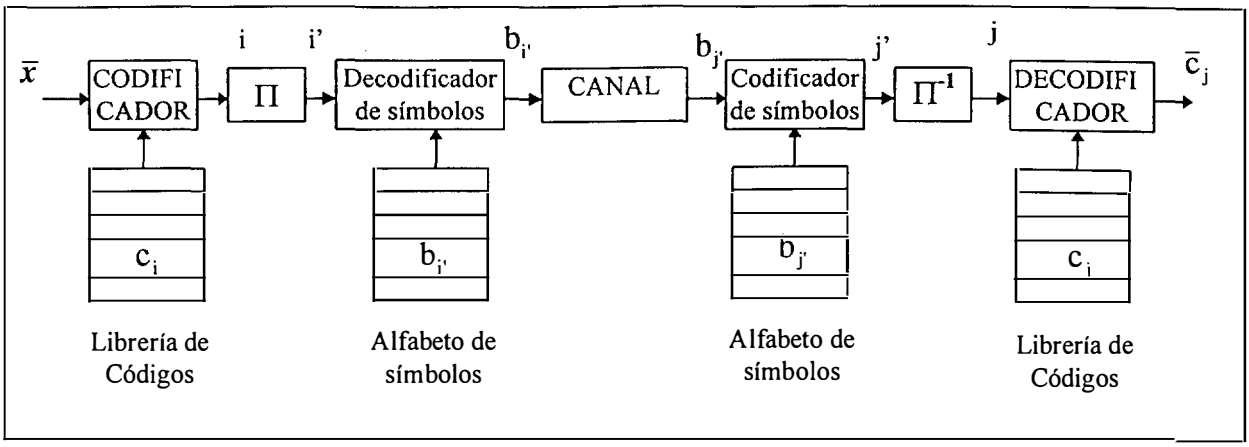

Fig. 2.22 Sistema de Cuantificación Vectorial en presencia de errores de canal.

Por claridad notacional, de ahora en adelante se eliminará la tilde de i' y j' quedando siempre implícita la dependencia con la función de asignación Π de los términos siguientes:

$$\begin{aligned} b_{i'} &\rightarrow b_i \\ b_{j'} &\rightarrow b_j \\ i' &\rightarrow i \\ j' &\rightarrow j \\ \bar{c}_{j'} &\rightarrow \bar{c}_j \end{aligned} \qquad \textbf{(2.1.4.1.8)}$$

Cuando sea conveniente, no obstante, se volverá a la notación con tildes.

2.1.4.2- Medida de la distorsión

Similar a la expresión (2.1.3.1.7) la distorsión media del proceso es, de nuevo, la esperanza del error cuadrático medio entre la entrada y la salida:

$$D = E\left\{\left\|\bar{x} - \bar{c}_j\right\|_2\right\} \qquad \textbf{(2.1.4.2.1)}$$

Antes de obtener una expresión para D en función de los parámetros del sistema, es preciso reparar sobre los dos procesos aleatorios presentes en éste, que supondremos independientes entre sí, dado su distinto origen: por una parte, la entrada $\bar{x}$ caracterizada por su f.d.p. y por otro, los posibles errores en la transmisión de símbolos. Estos pueden ser caracterizados por el juego de probabilidades:

$$P_{ji} = \text{Prob}\left(b_j / b_i\right) \qquad \begin{matrix} i = 0, \ , L-1 \\ j = 0, \ , L-1 \end{matrix} \qquad \textbf{(2.1.4.2.2)}$$

que indican la probabilidad de que el símbolo b_j sea recibido cuando se transmitió el b_i, para todos los posibles símbolos en transmisión y recepción.

Teniendo en cuenta sólo el segundo de estos componentes aleatorios, la distorsión resultante, cuando a la entrada se tenga el vector genérico $\bar{x}$, perteneciente a la región genérica S_i será:

$$d_{xi} = \sum_{j=0}^{L-1} P_{ji} \left\| \bar{x} - \bar{c}_j \right\|_2 d\bar{x} \qquad \textbf{(2.1.4.2.3)}$$

integrando esta medida de distorsión "individual" a lo largo de toda la extensión de S_i, se tendrá la distorsión asociada a esta región:

$$D_i = \int_{S_i} f_X(\bar{x})\, d_{xi}\, d\bar{x} = \int_{S_i} f_X(\bar{x}) \sum_{j=0}^{L-1} P_{ji} \left\| \bar{x} - \bar{c}_j \right\|_2 d\bar{x} \qquad \textbf{(2.1.4.2.4)}$$

Por último, la distorsión global buscada resulta de sumar las distorsiones parciales de las L regiones S_i:

$$D = \sum_{i=0}^{L-1} D_i = \sum_{i=0}^{L-1} \int_{S_i} f_X(\bar{x}) \sum_{j=0}^{L-1} P_{ji} \left\| \bar{x} - \bar{c}_j \right\|_2 d\bar{x} \qquad \textbf{(2.1.4.2.5)}$$

Obsevando esta expresión, hay que hacer notar que la distorsión, además de ser función de la distribución probabilística de la fuente ($f_x(\bar{x})$), del libro de códigos empleado ($\bar{c}_j$) y de la regla de codificación (S_i), también depende de la distribución de los errores de canal y de la función de asignación Π (plasmados ambos en los parámetros P_{ji}).

De (2.1.4.2.5), sin mucha dificultad, se puede llegar a expresar esta distorsión como composición de dos términos, uno ligado al proceso de cuantificación y otro a los errores que introduce el canal [Farvardin 90]:

$$D = \sum_{i=0}^{L-1} \int_{S_i} f_X(\bar{x}) \left\| \bar{x} - \hat{c}_i \right\|_2 d\bar{x} + \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} P_i P_{ji} \left\| \bar{c}_{ij} - \hat{c}_i \right\|_2 \qquad \textbf{(2.1.4.2.6)}$$

siendo $\hat{c}_i$ el centroide (centro de masas) de la región S_i:

$$\hat{c}_i = \frac{\int_{S_i} \bar{x}\, f_X(\bar{x})\, d\bar{x}}{\int_{S_i} f_X(\bar{x})\, d\bar{x}} \qquad \textbf{(2.1.4.2.7)}$$

y P_i la probabilidad de que el vector $\bar{x}$ se encuentre en esta región:

$$P_i = \int_{S_i} f_X(\bar{x})\, d\bar{x} \qquad \textbf{(2.1.4.2.8)}$$

Si el decodificador del cuantificador vectorial atiende a la Regla de los Centroides (la obtenida para el caso no ruidoso en (2.1.3.2.2.1)), entonces,

$$\bar{c}_i = \hat{c}_i \qquad i \quad 0, \quad , L-1 \qquad \textbf{(2.1.4.2.9)}$$

con lo que el primer sumando de (2.1.4.2.6) se corresponde con la distorsión de fuente (la que habría en el sistema si no existieran errores de canal):

$$D_F = \sum_{i=0}^{L-1} \int_{S_i} f_X(\bar{x}) \left\| \bar{x} - \hat{c}_i \right\|_2 d\bar{x} \qquad \textbf{(2.1.4.2.10)}$$

mientras que el segundo sumando se corresponde con el ruido debido al canal (distorsión de canal):

$$D_C = \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} P_i P_{ji} \left\| \bar{c}_{ij} - \bar{c}_j \right\|_2 \qquad \textbf{(2.1.4.2.11)}$$

Se tiene, por tanto, que la distorsión global del sistema es suma de la distorsión de fuente y la de canal:

$$D \leq D_F + D_C \qquad \textbf{(2.1.4.2.12)}$$

No obstante, si las anteriores Reglas no están presentes, ambas distorsiones estarán interrelacionadas.

Sin embargo, cuando existe ruido en el canal, la anterior regla no resulta óptima. Como se explicaba en los Apartados 2.1.4.4 y 2.1.4.6, la Regla Generalizada de los Centroides era la óptima, no siendo equivalente, en general, a su versión "no ruidosa". Si la probabilidad de error en el canal es reducida ambas reglas son próximas. En cualquier caso, incluso sin estar presente ninguna de las anteriores reglas, la distorsión total del sistema está acotada por la suma de la distorsión de fuente y la distorsión de canal [Zeger 87]:

$$D \leq D_F + D_C \qquad \textbf{(2.1.4.2.13)}$$

2.1.4.3- Caracterización del canal

Un canal ruidoso y sin memoria, se caracteriza por las probabilidades P_{ji} descritas en el apartado anterior.

Considerando que los símbolos son binarios y que el tamaño de la librería de códigos L es potencia de 2:

$$L=2^M \qquad \textbf{(2.1.4.3.1)}$$

sin pérdida de generalidad, se puede convenir que el alfabeto B de símbolos siga la ordenación natural:

$$\begin{aligned} b_0 &= \{00..00\} \\ b_1 &= \{00..01\} \\ &\\ b_{L-1} &= \underbrace{\{11..11\}}_{M \text{ bits}} \end{aligned} \qquad \textbf{(2.1.4.3.2)}$$

Por su parte, asumiremos que el canal es simétrico, quedando su comportamiento completamente descrito por la probabilidad de error de un bit (PEB) de magnitud ε:

$$PEB = Prob\{'1' \rightarrow '0'\} = Prob\{'0' \rightarrow '1'\} = \varepsilon \qquad \textbf{(2.1.4.3.3)}$$

Según todo lo anterior, la probabilidad P_{ji} vendrá dada por [Farvardin 90]:

$$P_{ji} = Prob\left(b_j / b_i\right) = (1-\varepsilon)^{M-\|b_j, b_i\|_{ham}} \cdot \varepsilon^{\|b_j, b_i\|_{ham}} \qquad \textbf{(2.1.4.3.4)}$$

siendo $\|b_1, b_2\|_{ham}$ la distancia Hamming entre los binarios de igual longitud b_1 y b_2 dada por el número de bits diferentes que tienen.

2.1.4.4- Condiciones de optimalidad

Para un canal dado, la distorsión media del proceso, según las explicaciones de las secciones anteriores, es función de la regla de codificación dada por la partición $\{S\}$, del decodificador dado por el libro de códigos $\{C\}$ y de la función de asignación Π.

Existen condiciones de optimalidad para $\{S\}$ y para $\{C\}$ semejantes a las del caso no ruidoso, que veremos a continuación. Sin embargo, aquí tampoco existe un procedimiento global analítico para optimizar D.

2.1.4.4.1- Regla Generalizada del Vecino Más Próximo (RGVMP)

Para un libro de códigos $\{C\}$ y una asignación Π dados, se obtiene la partición $\{S\}$ óptima con las mismas consideraciones que para el caso escalar o el vectorial no ruidoso: la entrada $\bar{x}$ se asignará al vector código que conduzca a la mínima distorsión. Si el codificador elige el índice i (el vector código elegido es $\bar{c}_i$), la distorsión tendrá una magnitud dada por:

$$\sum_{j=0}^{L-1} P_{ji} \left\| \bar{x} - \bar{c}_j \right\|_2 \qquad \textbf{(2.1.4.4.1.1)}$$

Se pretende que esta distorsión sea siempre menor o igual que la que pudiera haberse obtenido si $\bar{x}$ hubiera sido codificado en cualquier otro índice:

$$\sum_{j=0}^{L-1} P_{ji} \left\| \bar{x} - \bar{c}_j \right\|_2 \leq \sum_{j=0}^{L-1} P_{jp} \left\| \bar{x} - \bar{c}_j \right\|_2 \qquad \forall\, p = 0,\ 1,..,\ L-1. \qquad \textbf{(2.1.4.4.1.2)}$$

Para que esta desigualdad se cumpla siempre, basta definir las subparticiones en función de los vectores código, de acuerdo con:

$$S_i = \left\{ \bar{x} \mid \sum_{j=0}^{L-1} P_{ji} \left\| \bar{x} - \bar{c}_j \right\|_2 \leq \sum_{j=0}^{L-1} P_{jp} \left\| \bar{x} - \bar{c}_j \right\|_2 ,\ \forall\, p = 0,\ 1,..,\ L-1. \right\} \qquad \textbf{(2.1.4.4.1.3)}$$

2.1.4.4.2- Regla Generalizada de los Centroides (RGC)

Considerando fijos la partición $\{S\}$ y la función de asignación Π. También mediante un desarrollo matemático sencillo se puede llegar a que los centroides óptimos, aquéllos que conducen a una mínima distorsión D, están dados por [Miller 94]:

$$\bar{c}_i \quad \frac{\sum_{j=0}^{L-1} P_{ji} \cdot P_j \cdot \hat{c}_j}{\sum_{j=0}^{L-1} P_{ji} \cdot P_j} \qquad i \ 0,1, \ L\text{-}1 \qquad \textbf{(2.1.4.4.2.1)}$$

donde P_i es la probabilidad dada en (2.1.4.2.8) de que el vector $\bar{x}$ se encuentre en la región S_i, P_{ji} es la probabilidad dada en (2.1.4.3.4) de que el símbolo enviado b_i sea recibido como el b_j y $\hat{c}_j$ es el centroide de la región S_i expresado en (2.1.4.2.7).

2.1.4.5- El Algoritmo de Lloyd Generalizado (GLA)

Este algoritmo, extensión al caso ruidoso del algoritmo LBG, es debido a H. Kumazawa [Kumazawa 84] y también puede encontrase en [Farvardin 90] y [Miller 94].

Al igual que con el algoritmo de Lloyd y con el LBG, las dos reglas de optimalidad, ahora en sus versiones generalizadas, son aplicadas alternada e iterativamente.

Llamando iteración GLA al bucle básico consistente en aplicar las nuevas reglas RGVMP y RGC (Figura 2.23), se puede representar el funcionamiento del algoritmo completo, según los diagramas de la Figura 2.24.

Igualmente el algoritmo se detendrá cuando el descenso de la distorsión relativa

$$D_r[m] = \frac{D[m-1] - D[m]}{D[m]} \qquad \textbf{(2.1.4.5.1)}$$

en una iteración del bucle no supere un umbral mínimo.

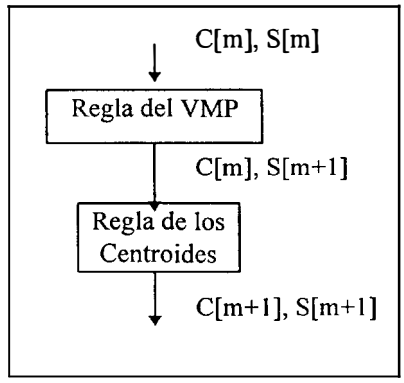

Fig. 2.23 Iteración GLA

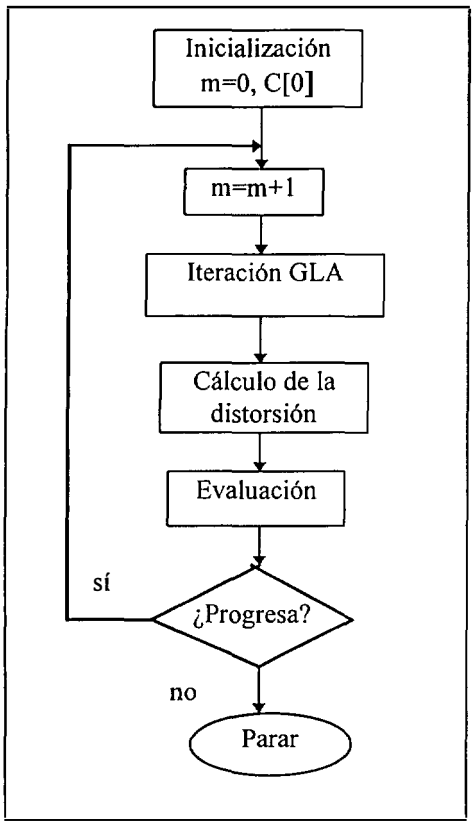

Fig. 2.24.a Algoritmo GLA (diagrama de flujo)

Paso 1. Inicialización del libro de códigos
Paso 2. Aplicación de la RGVMP
Paso 3. Aplicación de la RGC
Paso 4. SI la distorsión se reduce,
saltar al Paso 2
Paso 5. SI NO,
parar

Fig. 2.24.b Algoritmo GLA (pseudocódigo)

2.1.4.6- Diseño basado en datos empíricos

Nuevamente, como la descripción analítica de la f.d.p. del vector $\bar{x}$ no suele ser frecuente, se recurre a utilizar un conjunto de entrenamiento

$$V = \{\bar{v}_0, \bar{v}_1, .., \bar{v}_{N-1}\} \qquad \textbf{(2.1.4.6.1)}$$

que represente estadísticamente la distribución de $\bar{x}$.

Paralelos a los desarrollos del caso escalar y del vectorial sin errores de canal encontramos las nuevas formulaciones para la distorsión media del proceso y las reglas RGVMP y RGC. La distorsión media quedará:

$$D = \sum_{p=0}^{N-1} \left\| \bar{v}_p - Cod(\bar{v}_p) \right\|_2 \qquad \textbf{(2.1.4.6.2)}$$

Llamando V_i al subconjunto de cardinal N_i de V con aquellos vectores que son codificados con el índice i, esto es:

$$V_i = \{\bar{v} \in V / Cod(\bar{v}) = i\} = \{\bar{v}_{ik}, k = 0, 1.., N_i - 1\} \qquad \textbf{(2.1.4.6.3)}$$

se tendrá que la distorsión media dada en (2.1.4.6.2) podrá ponerse como la suma de las distorsiones parciales D_i asociadas a cada región:

$$D = \sum_{i=0}^{L-1} D_i = \frac{1}{N} \sum_{i=0}^{L-1} \sum_{k=0}^{N_i-1} \sum_{j=0}^{L-1} P_{ji} \left\| \bar{v}_{ik} - \bar{c}_j \right\|_2 \qquad \textbf{(2.1.4.6.4)}$$

La Regla del Vecino Más Próximo se formulará ahora según:

$$Cod(\bar{v}) = i \Leftrightarrow \sum_{j=0}^{L-1} P_{ji} \left\| \bar{x} - \bar{c}_j \right\|_2 \leq \sum_{j=0}^{L-1} P_{jp} \left\| \bar{x} - \bar{c}_j \right\|_2 \quad \forall\, p = 0,\ 1,..,\ L-1$$

(2.1.4.6.5)

lo cual implica que la partición $\{V\}$ del conjunto de entrenamiento V consistirá en los subconjuntos V_i dados por:

$$V_i = \left\{ \bar{v} \mid \sum_{j=0}^{L-1} P_{ji} \left\| \bar{v} - \bar{c}_j \right\|_2 \leq \sum_{j=0}^{L-1} P_{jp} \left\| \bar{v} - \bar{c}_j \right\|_2 ,\ \forall\, p = 0,\ 1,..,\ L-1. \right\}$$

(2.1.4.6.6)

La Regla de los Centroides quedará igual que en (2.1.4.4.2.1):

$$\bar{c}_i = \frac{\sum_{j=0}^{L-1} P_{ji} \cdot P_j \cdot \hat{c}_j}{\sum_{j=0}^{L-1} P_{ji} \cdot P_j} \qquad i\ 0,\ 1,\ L\text{-}1 \qquad \textbf{(2.1.4.6.7)}$$

sólo que ahora la probabilidad P_j de que el vector $\bar{v}$ se encuentre en la región V_j estará dada por:

$$P_j = \frac{N_j}{N} \qquad j\ 0,\ 1,\ L\text{-}1 \qquad \textbf{(2.1.4.6.8)}$$

El problema de que alguna de estas regiones se encuentre vacía no tiene la gravedad que en el caso no ruidoso, de donde surgía una indeterminación en el cálculo de los centroides según (2.1.3.4.7), circunstancia que no ocurrirá con su versión generalizada dada en (2.1.4.6.7) y (2.1.4.6.8).

En cualquier caso, no tiene mucho sentido tener regiones vacías, por lo que éstas se suelen eliminar con el mismo procedimiento que se explicaba en la sección 2.1.3.4 para el caso no ruidoso.

El cuanto al algoritmo GLA, éste podrá aplicarse en el caso empírico tal y como se describe en el apartado 2.1.4.5 teniendo en cuenta la nueva formulación de las reglas de optimalidad y de la distorsión.

2.2- Técnicas Evolutivas

2.2.1- Introducción

Queremos recoger bajo este nombre el gran abanico de técnicas heurísticas de optimización basadas en la evolución de una población de individuos (tentativas soluciones al problema planteado) sometidas a diversas leyes de selección y alteración. Estos procedimientos se articulan de tal modo que los individuos que suponen mejores soluciones al problema son elegidos y los peores eliminados. Más tarde los primeros son sometidos a uno o varios procesos de transformación que dan lugar a la creación de nuevos individuos cuyas características proceden en parte de sus antecesores y en parte son incorporadas de forma aleatoria. El proceso continúa con nuevas y sucesivas selecciones y alteraciones que conducirán, previsiblemente, a individuos muy adaptados, esto es, a soluciones de buenas prestaciones en el problema.

La relación de estos procedimientos con las ideas darwinianas de la Selección Natural y la Evolución de las Especies es evidente. En realidad estos métodos tratan de emular el simple, elegante, cruel e incomparablemente eficaz mecanismo por el cual las especies naturales han ido perfeccionándose, adecuándose a su entorno a lo largo de su Historia. Este "juego de la "vida" no es otro que el "ensayo y error biológico" en el que la función de optimización es la supervivencia de las especies, según la escolástica tradicional darwiniana [Darwin 76] o la de los genes, atendiendo a las nuevas formulaciones genetistas [Dawkins 93].

Si la Evolución Natural ha logrado individuos, especies y genes adecuados a entornos variadísimos, cambiantes y a menudo hostiles, esto es, soluciones de enorme grado de sofisticación y perfección ante problemas de semejante magnitud, quizás sea también válida, aún tan sólo en sus líneas maestras, para dar solución a diversos problemas que se plantean en la Ingeniería o en otras ramas del saber y que al hombre le resultan complicados. Este argumento, al lado de las capacidades computacionales cada vez mayores de los ordenadores parece haber desencadenado en las últimas tres décadas el surgimiento de una pléyade de técnicas basadas en las anteriores ideas, que resuelven un sinfín de problemas de optimización y que aparecen en la literatura con diversos nombres: Algoritmos Genéticos [Holland 75], Estrategias Evolutivas [Schewefel 81], Programas Evolutivos [Michalewicz 92], Evolución Simulada [Fogel 91], Métodos Darwinianos [Zhang 93], o Algoritmos Evolutivos [Hoffmeister 91].

En general, la diferencia entre unos y otros se encuentra en la forma de representar a los individuos, los mecanismos de selección, y procreación y la incorporación o no de optimizadores locales. En las secciones siguientes se describen algunas de las alternativas más frecuentes para estos elementos.

Entre los mecanismos de procreación, merecen mención especial dos grupos fundamentales: los de cruzamiento o cruce, que crean uno o varios individuos nuevos (hijos) a partir de individuos existentes (padres); y los de mutación que operan de forma individual sobre cada individuo, variando sus características en mayor o menor grado.

No se puede concluir esta Introducción sin presentar a un componente fundamental en todas las técnicas evolutivas, implícito en varios de los mecanismos antes enunciados: la aleatoriedad. Estas técnicas, incluida la Evolución Natural de las Especies, confían en la ejecución no determinística de los procesos. Si la generación de

varios individuos nuevos a partir de dos padres siempre diera lugar a hijos idénticos, se mermaría la capacidad exploratoria al reducir el número de observaciones diferentes en el espacio de búsqueda. Al incluir aleatoriedad en el cruce se asegura que los hijos tengan gran probabilidad de ser diferentes, eliminando la contrariedad anterior.

Por otro lado, el operador mutación constituye en esencia un "difusor" de movimiento aleatorio entre los miembros de una población, también con el objeto de "saltar" a posiciones nuevas en la búsqueda, complementando al operador de cruce, según se explicará más adelante.

Generalmente también se dota de aleatoriedad a los procesos de selección. El argumento es similar: permitiendo que los individuos peor dotados tengan alguna probabilidad de ser seleccionados se conseguirá que la búsqueda tenga un carácter más global y flexible posibilitando que alguno de los caminos de exploración sea en algunos momentos poco propicio. Basta el símil del fértil valle situado tras la montaña; para llegar a él (solución adecuada al problema) a partir de una altiplanicie (solución intermedia) será preciso recorrer la montaña (solución inadecuada).

2.2.2- Formulación básica

Sea x una solución genérica perteneciente al espacio S de posibles soluciones al problema de optimizar la función f(x) y sea:

$$P(t) = \{x_1(t), x_2(t), .., x_T(t)\} \qquad \textbf{(2.2.2.1)}$$

una población de T individuos que progresan a lo largo de las generaciones t.

En un programa evolutivo genérico, una vez generada una población inicial P(0), se la someterá repetidas veces a los procesos de evaluación, selección y procreación hasta que alguna condición de finalización sea satisfecha. El esquema básico es el de la Figura 2.25.

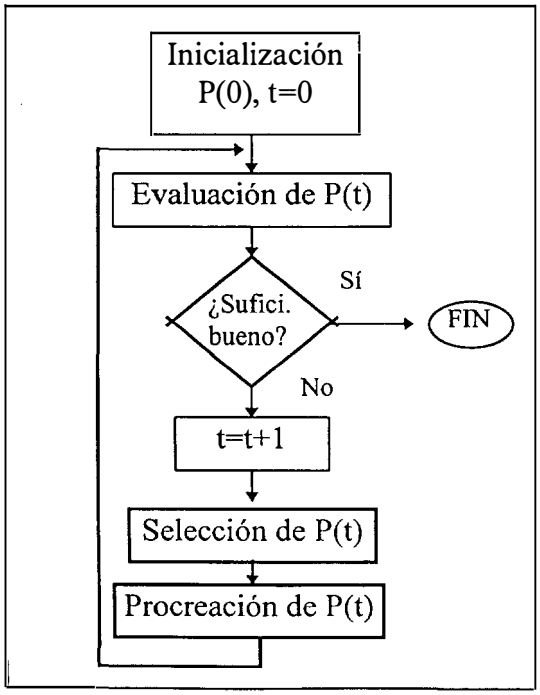

Fig. 2.25.a Programa evolutivo (diagrama de funcional).

Paso 1. Inicialización.
Paso 2. Evaluación de individuos (libros de código tentativos).

Paso 3. Sl se ha llegado a la convergencia o ya se han sucedido un cierto número de generaciones, PARAR.

Paso 4. Seleccionar los individuos más adaptados (padres).

Paso 5. Procrear nuevos individuos (hijos).

Paso 6. VOLVER al Paso 2.

Fig. 2.25.b Programa evolutivo (pseudocódigo).

2.2.3- Representación de individuos

En general la variable x podrá contener varias componentes (variables unidimensionales continuas o discretas) estructuradas de forma diversa: escalares, vectores, matrices, listas, grafos, etc. En principio, lo más natural es que los individuos tomen la representación que originalmente se le da a x.

Sin embargo, para la aplicación de los operadores de procreación y selección, a veces puede ser ventajoso que adopte otra representación. Veamos las opciones más corrientes:

a- Representación binaria

Una de las representaciones más habituales que se da en las técnicas evolutivas es la codificación binaria, según la cual todas las componentes de x se expresan en forma de cadenas de bits (símbolos binarios que pueden tomar los valores 0 ó 1). Si, por ejemplo, x consta de J componentes:

$$x = \{x_0, x_1, .. x_{J-1} \qquad \textbf{(2.2.3.1)}$$

a cada una de ellas le será asignado un número M_i de bits:

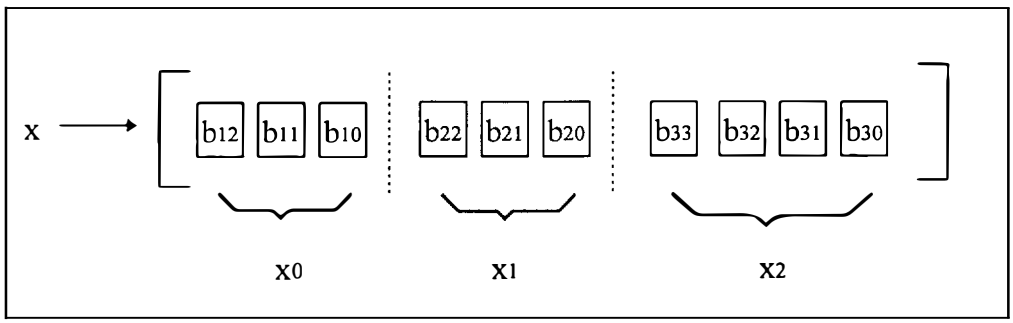

Fig. 2.26 Represnetación binaria de individuos

Como se esquematiza en la figura, la variable x quedará representada por la ristra de $M = \sum_{i=0}^{J-1} M_i$ bits resultantes de concatenar las representaciones binarias de las componentes x_i unas a continuación de las otras.

Si estas x_i son variables continuas, implícitamente se está considerando una cuantificación sobre las mismas, por el hecho de que ser expresadas con un número finito de bits. Así pues, si la componente x_i toma valores en el intervalo $[x_{ii}, x_{is}]$ (el subíndice "i" se refiere a "inferior" y el "s" a "superior"), sólo se podrán expresar 2^{M_i} puntos diferentes en este intervalo, lo que significa que el tamaño de los intervalos y saltos de cuantificación (asumiendo cuantificación uniforme) será de:

$$\Delta_i = \frac{x_{is} - x_{ii}}{2^{M_i}} \qquad \textbf{(2.2.3.2)}$$

o mejor,

$$\Delta_i = \frac{x_{is} - x_{ii}}{2^{M_i} - 1} \qquad \textbf{(2.2.3.3)}$$

si queremos que los puntos x_{ii} y x_{is} sean incluidos en los valores de cuantificación.

El éxito de la representación binaria de los datos ha ido estrechamente unido al de una de las técnicas evolutivas de mayor profusión: los Algoritmos Genéticos (AG) [Holland 75], [Goldberg89]. En realidad, junto a dos operadores de procreación, la mutación y el cruce genéticos, que luego veremos, la representación binaria constituye uno de los elementos definitorios de los AG en el conjunto de las técnicas evolutivas.[1]

La representación binaria de los datos en los AG está estrechamente vinculada a los anteriores operadores y es, en esencia una simplificación de la dicotomía en el mundo natural entre individuos y genes. La información genética de los organismos vivos, ya sean animales, plantas, bacterias o virus está contenida en los cromosomas presentes en cada una de sus células.[2] Estos constituyen los planos de la construcción y funcionamiento del individuo, las complejísimas indicaciones de cómo éste ha de crecer y comportarse. Existen, por tanto, dos perspectivas o contextos: uno el del propio individuo, su estructura corporal y su forma de relacionarse con el entorno (fenotipo); otro, el de su representación, consistente en los mencionados planos, los cromosomas (genotipo).

El hecho de la creación de un ser a partir de otro u otros de la misma especie, piedra angular del ciclo de la vida, se efectúa principalmente en el contexto de los genes. En el momento de concebirlo, lo que los padres entregan físicamente al hijo son sus planos genéticos (los del hijo), engendrados a partir de una "barajada" combinación de los suyos (los de los padres), no le dan partes de sus cuerpos, sus propios ojos o brazos.

Por otra parte, las alteraciones aleatorias que surgen de forma esporádica en los individuos de una especie, para que tengan alguna significación evolutiva, han de realizarse en el contexto genético, pues si no, no pasarán a las siguientes generaciones y se extinguirán irremisiblemente.[3] Las alteraciones esporádicas en los contenidos genéticos de los individuos se llaman mutaciones genéticas y corresponden a "errores de copia" de los planos cromosómicos que los padres entregan a los hijos.

Como ya se ha apuntado, los AG también operan en un doble plano, en el que los individuos se expresan y son modificados a partir de una representación binaria, aún cuando corresponden a las variables objeto de la optimización, que toman valores en otros dominios.

En el mundo natural, todas las células de un individuo contienen idénticas copias de sus cromosomas. El número de cromosomas en una especie concreta es fijo (46 en el hombre), aunque varía de especie en especie. Cada cromosoma está formado por una molécula de ADN constituida, a su vez, por un par de cadenas de nucleótidos enrolladas en espiral. Los nucleótidos son pequeñas moléculas orgánicas de cuatro tipos distintos: Adenina, Guanina, Citosina y Timina. La sucesión de estos elementos determina los

[1] En realidad, la formulación de Holland es un poco más general, incluyendo representaciones de símbolos de reducido cardinal. Por otra parte, también se suelen considerar genéticas las representaciones mediante cadenas de enteros en los problemas de optimización combinatoria.

[2] El caso particular de los virus será sucintamente explicado más adelante, al hilo de dos operadores genéticos avanzados inspirados en los virus y su supuesto papel en la evolución de otras especies.

[3] En este respecto, el hombre, con su cultura e Historia, sí que representa una excepción, puesto que es capaz de propagar sus conocimientos a generaciones venideras por medios no genéticos.

distintos genes. Por tanto, se puede decir que la información genética de un individuo está codificada en símbolos cuaternarios.

Sin embargo, en este punto los AG no han seguido la analogía con la Naturaleza incorporando la más sencilla simbología binaria. Esta elección ha estado condicionada probablemente por la madurez de la Electrónica Digital y de la Lógica Computacional, ambas cimentadas sobre codificaciones y operaciones binarias (físicas o lógicas). En cualquier caso, lo cierto es que esta representación es la que presenta la información de una variable desglosada en un mayor número de elementos sobre los que pueden actuar los operadores de procreación, que más tarde comentaremos.

Es más, la evaluación de una población de M individuos representados de forma binaria puede dar mucha más información sobre la superficie inspeccionada que la que se obtendría usando otra codificaciones, por ejemplo, en punto flotante.

Pongamos un ejemplo. Supongamos que se quiere optimizar la función

$$f(x) = 1 - \cos\left(2\pi \frac{x}{100}\right) \qquad \textbf{(2.2.3.4)}$$

en el intervalo [10,80] y que se dispone de individuos (cromosomas) de 5 bits, que expresaremos con la notación:

$$x \rightarrow B_x = [b_4, b_3, b_2, b_1, b_0] \qquad \textbf{(2.2.3.5)}$$

Suponiendo una cuantificación uniforme del espacio de búsqueda y que se incluyen los extremos 10 y 80 en el conjunto de valores cuantificados, la resolución (diferencia entre dos valores cuantificados contiguos) vendrá dada por:

$$\Delta = \frac{80 - 10}{2^5 - 1} = 2.2581 \qquad \textbf{(2.2.3.6)}$$

y la relación entre los bits b_i y el valor de x:

$$x = 10 + \Delta \cdot \sum_{i=0}^{4} 2^i \, b_i \qquad \textbf{(2.2.3.7)}$$

Imaginemos que una población está formada por 4 individuos (puntos) con los siguientes valores:

x (individuos)	R_x (Representación real)	B_x (Representación binaria)	B_e (Representación entera)	f(x) (Función de adaptación)
x_1	39.3548	0 1 1 0 1	13	0.3799
x_2	64.1935	1 1 0 0 0	24	1.7782
x_3	28.0645	0 1 0 0 0	8	0.0185
x_4	52.9032	1 0 0 1 1	19	1.1814

Tabla 2.1

Se puede ver que x_2 y x_4 son soluciones mucho mejores que x_1 y x_3, siendo esta última particularmente poco adecuada. Esta información es accesible para ambas

representaciones. En realidad, es independiente del tipo de representación de los datos, pues sólo depende de f(x) y, por tanto, de x.

Sin embargo, el papel marcadamente relevante del bit b_4 o el carácter de menor importancia del bit b_0 es algo particularmente accesible a una representación binaria, como la aquí descrita.

En este momento resulta interesante introducir el concepto de patrón de similitud o esquema, como lo denomina Holland [Holland 75]. Este es un subconjunto de todos los individuos posibles que tienen uno o varios de sus bits con valores fijos. Por ejemplo, 1*0** es el esquema de todos los individuos cuyo bit b_4 es 1 y b_2, el 0.

Con la evaluación en paralelo de individuos representados de forma binaria, no sólo se procesan (evalúan) individuos, sino también esquemas. El número de esquemas procesados en cada generación del AG es del orden de M^3 [Goldberg 89]. Este sorprendente resultado, tan favorable si se tiene en cuenta que el número de individuos evaluados es de M, se le conoce con el nombre de "Paralelismo Implícito".

A pesar de la elegante propuesta de la representación binaria de los datos, de sus sólidos y sencillos fundamentos matemáticos ("Paralelismo Implícito, Teoría del Esquema", etc. [Holland 75], [Goldberg 89]), de su facilidad para ser creados y manipulados, y de su carácter general y robusto, presenta importantes carencias a la hora de resolver problemas de cierta complejidad. Como relata Michalewicz: "las representaciones binarias no son siempre apropiadas para problemas altamente constreñidos y otras representaciones son frecuentemente más naturales" [Michalewicz 93, pg.53].

En la misma línea se pronuncia Davis, quien defiende el uso de representaciones específicas en el dominio concreto de la aplicación [Davis 91].

En el caso de la presente Tesis también resulta más natural la representación en punto flotante de los datos, como se verá más adelante.

b- Representación mediante listas de enteros

En algunos tipos importantes de problemas combinatorios se trata de encontrar la mejor ordenación posible de una lista finita de elementos conocidos. Tal es caso del conocido *Problema del Viajante de Comercio* (*Traveling Salesman Problem* en terminología sajona, o más sencillamente TSP). Consiste en que un agente de comercio tiene que atravesar un conjunto de N ciudades con el propósito de recorrer la mínima distancia posible, pasando por todas ellas únicamente una vez.

Nombrando a las ciudades con un número entero de la 1 a la N, las distintas soluciones al problema lo constituyen las N! permutaciones posibles de la lista $\{1, 2, .., N\}$. La lista

$$\{i_1, i_2, .., 1_N\} \qquad \textbf{(2.2.3.8)}$$

expresa la solución según la cual el viajante visita primero la ciudad i_1, luego la i_2 y así hasta la i_N

Hay general consenso entre la comunidad especializada de que la representación binaria de este tipo de problemas es inadecuada [Whitley 89], debido principalmente al fuerte condicionamiento que impone la estructura de los datos. Es preferible y a la vez más natural, que éstos se expresen directamente como listas de enteros.

En otros problemas combinatorios como son, por ejemplo, los de "coloreado de gráficos", planificación y particionado [Michalewicz 92] también la representación entera suele ser más adecuada que la binaria.

2.2.4- Mecanismos de selección

La primera parte para el paso de una generación a la siguiente en las técnicas evolutivas es la selección de los mejores individuos para su posterior modificación o procreación. Sin embargo, la manera de seleccionar los individuos más adecuados es diversa. Veamos tres de los mecanismos más utilizados:

a- Selección Aleatoria Proporcional a la Función de Prestaciones [Goldberg 89]

Llamaremos función objetivo a la función f(x) que se trata de optimizar. La función de prestaciones F(x) o de "adecuación" ("fitness") debe indicar el grado de bondad de la solución y debe tomar valores siempre positivos y tanto mayores cuanto mayor sea f(x) en caso de maximización o cuanto menor sea f(x), en caso de minimización.

Así, por ejemplo, si se trata de maximizar la función f(x) y se sabe que ésta toma siempre valores positivos, puede hacerse que la función de prestaciones coincida con la función objetivo:

$$F(x)=f(x) \qquad \textbf{(2.2.4.1)}$$

Si, por el contrario, si se trata de minimizar f(x), se puede hacer que:

$$F(x) \;\; A-f(x) \quad \text{con } f(x) \leq A, \;\; \forall x \in S \qquad \textbf{(2.2.4.2)}$$

siendo A una cota superior de la función f(x) y S el espacio de las soluciones posibles.

En general, f(x) se podrá escalar de la forma:

$$F(x) \;\; C_1 f(x)+C_2 \qquad \textbf{(2.2.4.3)}$$

eligiendo las constantes C1 y C2 de tal forma que F(x) sea siempre sea mayor o igual a cero y que crezca cuando se trate de maximizar f(x) o decrezca cuando se quiera minimizar f(x).

De forma aún más general, se podrá aplicar a f(x) cualquier función g( ) monótona creciente (maximización) o decreciente (minimización):

$$F(x) \;\; G(f(x)) \qquad \textbf{(2.2.4.4)}$$

aunque esto no suele ser lo habitual.

Una vez elegida la función de prestaciones, la selección de un individuo de una población de tamaño T se efectúa de la siguiente manera:

1- Obtener la función de prestaciones de cada individuo de la población

$F(x_i)$, $i=1,2,\ T.$ **(2.2.4.5)**

2- Obtener la cantidad porcentual de "adecuación" correspondiente a cada individuo x_i:

$$x_i \rightarrow p_i = \frac{F(x_i)}{\sum_{k=1}^{T} F(x_k)} \qquad \textbf{(2.2.4.6)}$$

El conjunto de las p_i formará un esquema de probabilidades:

$$\begin{cases} 0 \le p_i \le 1 \\ \sum_{i=1}^{M} p_i = 1 \end{cases} \qquad \textbf{(2.2.4.7)}$$

3- Dividir el intervalo [0,1] en subintervalos concatenados de tamaño p_i como muestra el dibujo.

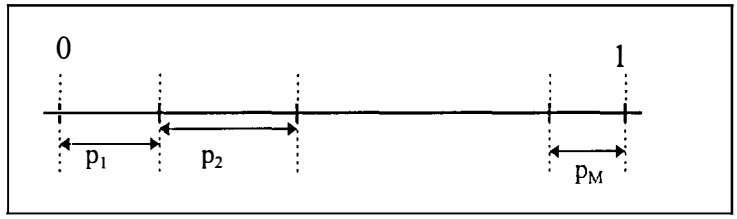

Fig. 2.26

4- Extraer una realización de una variable aleatoria uniformemente distribuida en [0,1].

5- Elegir el individuo del intervalo subintervalo i-ésimo donde se incluya el valor que ha tomado la variable aleatoria lanzada según 4.

De esta forma los individuos se eligen con una probabilidad más alta, cuanto mayor sea su función de adecuación, siguiendo las pautas de la selección natural darwiniana, por las que las especies más adaptadas al medio son las más aptas para la supervivencia.

b- Selección por Rango [Baker 85]

Cada individuo es seleccionado proporcionalmente al rango de su función objetivo y no a la magnitud que toma ésta. El algoritmo es el siguiente:

1- Se ordenan las T soluciones de $f(x_i)$ $(i=1,2,\ ,\ T)$ en orden de menor a mayor. Al mejor individuo se le da orden M y al peor, orden 1.

2- Se transforma linealmente este ordenamiento:

$g(x)$ A $r(x) + B$ **(2.2.4.8)**

donde A y B son constantes positivas a determinar y r(x) es el rango (orden) obtenido en 1.

3- Se aplica la selección aleatoria proporcional a las prestaciones sobre el resultado de 2.

La motivación de este mecanismo es doble: por un lado impide que en las primeras generaciones, unos pocos individuos de buenas prestaciones inunden completamente la población (problema del "superindividuo"). Por otra parte, en las últimas etapas del algoritmo, gran cantidad de individuos, incluidos los más adaptados, tienen prestaciones parecidas, por lo que su difusión en la población a través del sesgo de la selección es incierta. Al atender al rango y no a la función objetivo se sigue favoreciendo de forma sostenida a los individuos mejores, aun cuando la diferencia con el resto no sea muy notable.

c- Selección por Torneo [Goldberg 91]

Según este procedimiento, estudiado por Brindle [Brindle 81] se eligen al azar S individuos de la población y se selecciona el mejor de ellos.

Este último proceso de selección puede hacerse de forma probabilística proporcionalmente a la función de adecuación [Fogel 91].

2.2.5- Mecanismos de cruce

El cruce o entrecruzamiento se presenta como un operador de procreación de gran importancia en las técnicas evolutivas, en general, y en los AG, en particular.

Mediante el cruce, a partir de dos individuos (padres), se crea un individuo nuevo (hijo). Este tomará sus características (valores de sus componentes) del padre, de la madre, o de ambos de forma combinada, incluyendo siempre cierta aleatoriedad en estas elecciones. La razón de ser de este mecanismo, tanto en los AG como en la propia evolución natural es que la exploración en el espacio de representación (dominio de datos en los AG o acervo génico en el mundo natural) es mucho más rica y eficiente de esta manera, pudiéndose unir buenas características de ambos progenitores, obteniéndose en algunos casos individuos hijos mejores que los padres. En otros casos, se combinarán "malas" características de los padres, resultando individuos peor adaptados al entorno. La selección hará que los "buenos" individuos prosperen y los "malos" no, haciendo válido este mecanismo.

Normalmente en los AG la influencia de cada padre es simétrica, en plena consonancia con el mundo biológico en el que cada componente genético del cromosoma del hijo puede proceder del padre o de la madre con igual probabilidad, a pesar de que en las especies heterosexuales, tanto la estructura corporal como el comportamiento familiar y social del individuo esté fuertemente marcado por su sexo.

2.2.5.1- Cruce en cadenas binarias

Existen varias alternativas. Entre ellas destacan las siguientes:

a- Cruce por un Punto [Holland 75]

En los AG tradicionales, en los que los datos están representados por cadenas binarias (cromosomas) de tamaño fijo, el cruce entre dos individuos puede realizarse enfrentando a sus dos cromosomas, eligiendo al azar un punto de cruce y uniendo la parte derecha del primer cromosoma con la izquierda del segundo y viceversa, como se muestra en la Figura 2.27.

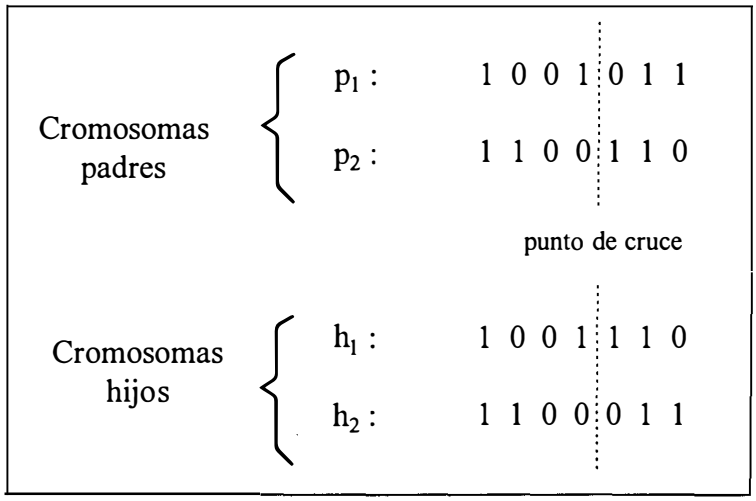

Figura 2.27 Mecanismo de Cruce por un Punto

b- Cruce Generalizado [De Jong 75]

Es similar al anterior, sólo que el número de puntos de cruce es arbitrario (Figura 2.28).

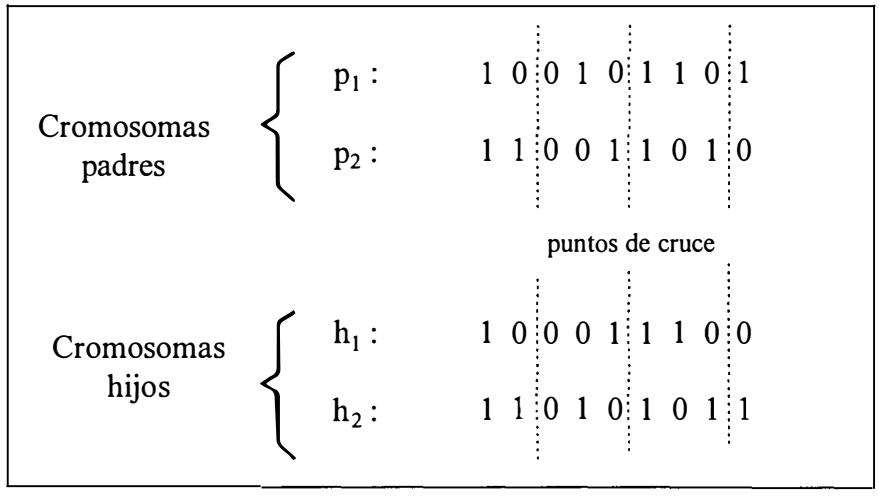

Figura 2.28 Mecanismo de Cruce Generalizado

c- Cruce Uniforme [Syswerda 89]

Con este operador también se generan dos hijos a partir de dos padres. Para cada posición de los bits se determina de forma aleatoria (probabilidad 1/2) si el padre p_1 ha de copiar su correspondiente bit en el hijo h_1 (y el padre p_2, su correspondiente bit en el hijo h_2) o viceversa.

En la figura 2.29 se presenta un ejemplo de la operación de este tipo de cruce, en el que la plantilla indica un 1 en las posiciones en las que el padre p_1 entregará su bit al hijo h_1 y el padre p_2 al hijo h_2.

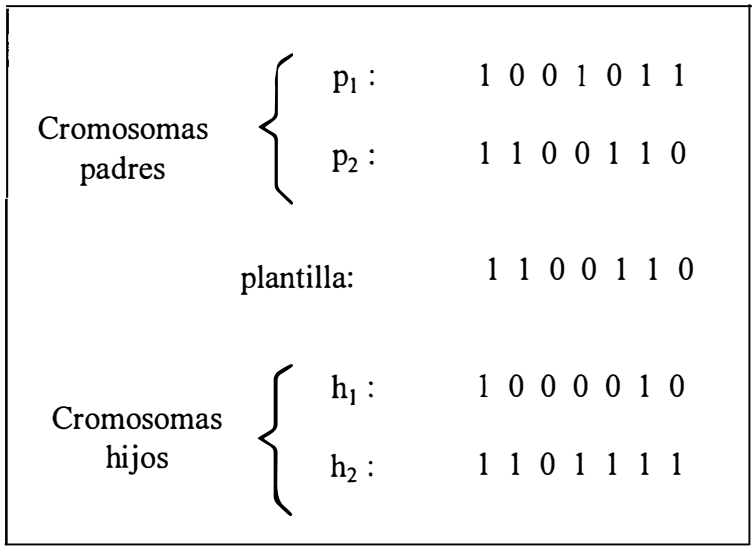

Figura 2.29 Mecanismo de Cruce Uniforme.

Puede verse que las características codificadas de forma compacta (bits próximos entre sí) son respetados en mayor medida con el cruce por un punto y el cruce generalizado que con este tipo de cruce. Por ello se dice que aquellos operadores son más locales que éste último.

2.2.5.2- Cruce en codificaciones reales

Sea $\overline{x}_i$ es un vector de J componentes reales:

$$\overline{x}_i = (\overline{x}_{i0}, \overline{x}_{i1}, .., \overline{x}_{iD-1}) \qquad \textbf{(2.2.5.2.1)}$$

a- El cruce por un Punto [Michalewicz 92] opera de forma idéntica al caso de codificación binaria, pero ahora lo que se permutan son componentes del vector en lugar de bits, como se aprecia en la Figura 2.30.

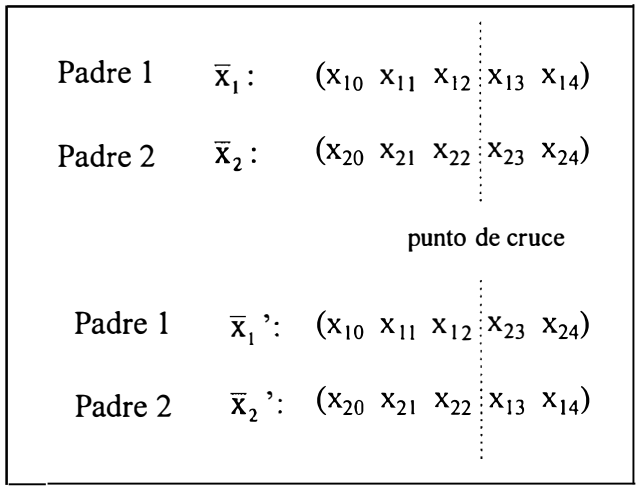

Figura 2.30 Mecanismo de cruce por un Punto en codificaciones reales

De manera análoga pueden concebirse operadores de cruce generalizado y cruce uniforme para vectores de componentes reales.

b- Cruce Aritmético [Michalewicz 92]

En este caso los vectores hijo serán combinaciones lineales de los dos vectores padre.

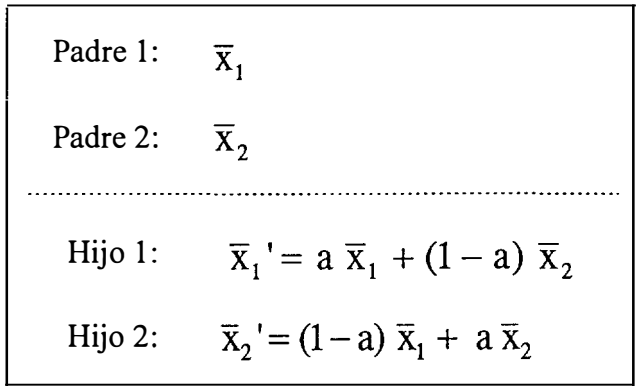

Figura 2.30 Mecanismo del Cruce Aritmétco

donde a es una constante entre 0 y 1. También es frecuente introducir aleatoriedad en este operador haciendo, por ejemplo, que a sea una variable aleatoria uniformemente distribuida en [0,1], o quizás con otra distribución, por ejemplo gaussiana [Zhang 93], [Malanda 96].

2.2.5.3- Cruce en listas de enteros

En este caso los elementos del espacio de búsqueda son permutaciones diferentes de una lista, según se describía en el Apartado 2.2.3.3, existiendo diversos operadores aplicables:

a- Cruce Parcialmente Mapeado [Goldberg 85]

El hijo se construye eligiendo segmentos de la lista de uno de los padres, preservando el orden y la posición del mayor número posible de elementos (ciudades en la terminología del TSP) del otro padre. Nos serviremos del siguiente ejemplo para explicar el procedimiento. El segmento de la lista del primer padre se forma eligiendo aleatoriamente los puntos de corte:

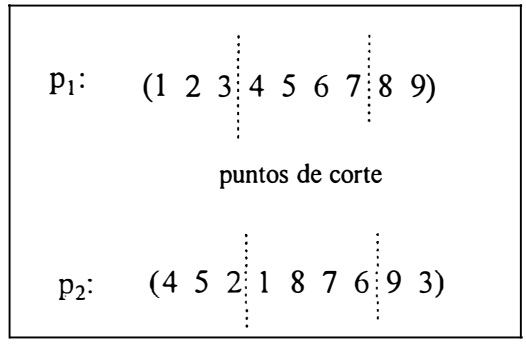

Figura 2.31

produciendo dos hijos, incompletos todavía, intercambiando sus segmentos centrales:

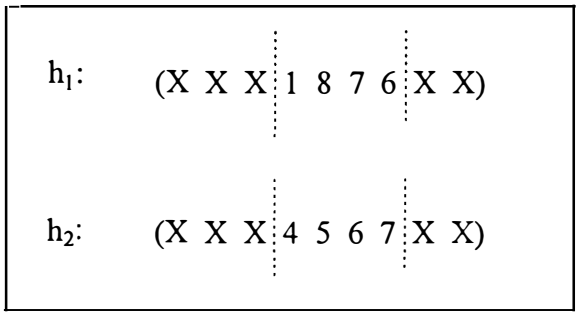

Figura 2.32

A su vez, este intercambio define las asociaciones siguientes:

$$1 \leftrightarrow 4$$
$$8 \leftrightarrow 5$$
$$7 \leftrightarrow 6 \qquad \textbf{(2.2.5.3.1)}$$
$$6 \leftrightarrow 7$$

Ahora se rellenarán las posiciones del primer hijo con los valores que tenían los padres, siempre que no haya ningún conflicto (que ninguno de estos valores quede repetido en la lista):

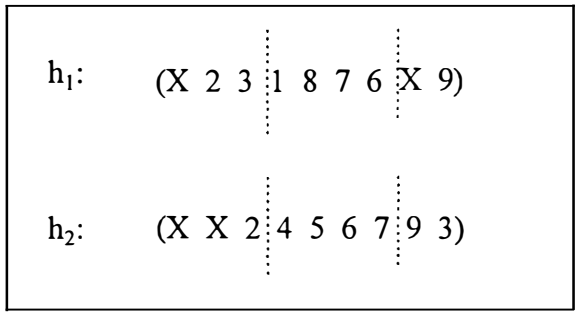

Figura 2.33

Finalmente a la primera X de h_1, que debería tomar el valor 1 si no tuviera conflicto, se le da el 4, atendiendo a la asociación dada en (2.2.5.3.1). De la misma forma se rellena el resto de las incógnitas, quedando:

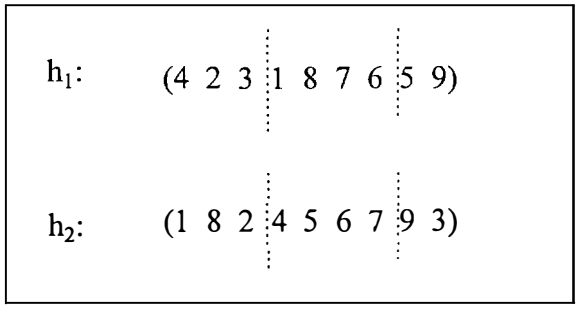

Fig. 2.34

b- Cruce Cíclico [Oliver 87]

En este caso se construyen los hijos de tal forma que cada posición (ciudad) proceda de uno de los padres. En el siguiente ejemplo, los padres:

$$p_1: \quad (1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9) \qquad \textbf{(2.2.5.3.2)}$$

$$p_2: \quad (4\ 1\ 2\ 8\ 7\ 6\ 9\ 3\ 5)$$

producirán el primer hijo tomando la primera ciudad del primer padre:

$$h_1 = (1\ x\ x\ x\ x\ x\ x\ x) \qquad \textbf{(2.2.5.3.3)}$$

Dado que todas las ciudades en el hijo deben tomarse a partir de uno de sus padres (en la misma posición), la siguiente ciudad que se ha de considerar es la 4 (primera ciudad del padre p_2). En p_1 esta ciudad está en la posición 4, por lo que

$$h_1 = (1\ x\ x\ 4\ x\ x\ x\ x) \qquad \textbf{(2.2.5.3.4)}$$

esto implica a la ciudad 8, por estar situada en p_2 justo debajo de la recién seleccionada 4 de p_1:

$$h_1 = (1\ x\ x\ 4\ x\ x\ x\ 8\ x) \qquad \textbf{(2.2.5.3.5)}$$

Siguiendo esta regla, las siguientes ciudades incluidas en el primer hijo serían la 3 y la 2. Sin embargo, la selección de la ciudad 2 requeriría la selección de la ciudad 1, que ya está en la lista, por lo que se completa el círculo:

$$h_1 = (1\ 2\ 3\ 4\ x\ x\ x\ 8\ x) \qquad \textbf{(2.2.5.3.6)}$$

el resto de las ciudades se toman del otro padre:

$$h_1 = (1\ 2\ 3\ 4\ 7\ 6\ 9\ 8\ 5) \qquad \textbf{(2.2.5.3.7)}$$

Cambiando el papel de los dos padres, se obtendrá el segundo de los hijos de forma similar:

$$h_2 = (4\ 1\ 2\ 8\ 5\ 6\ 7\ 3\ 9) \qquad \textbf{(2.2.5.3.8)}$$

Este operador preserva la posición absoluta de los elementos en las secuencias paternas.

c- Cruce Ordenado Uniforme [Davis 91]

Este operador preserva parte de la información de uno de los padres e incorpora información del otro, aunque aquí la información no está dada por los valores (ciudades) asociadas a una posición de la lista, sino por el orden de éstos. Veamos un ejemplo:

$$p_1: \quad (1\ 2\ 3\ 4\ 5\ 6\ 7\ 8) \qquad \textbf{(2.2.5.3.9)}$$

$$p_2: \quad (8\ 6\ 4\ 2\ 7\ 5\ 3\ 1)$$

- Se genera una lista binaria (máscara) del mismo tamaño que los padres:

m: (0 1 1 0 1 1 0 0) **(2.2.5.3.10)**

- Se copian al hijo 1 las ciudades del padre p_1 en las posiciones en las que la máscara presenta el valor 1:

h_1: (x 2 3 x 5 6 x x) **(2.2.5.3.11)**

- Se genera una lista con los elementos de p_1 donde la máscara está a 0.

L: (1 4 7 8) **(2.2.5.3.12)**

- Se permuta esta lista para que sus elementos aparezcan en el orden en que se encuentran en el padre p_2:

L': (8 4 7 1) **(2.2.5.3.13)**

- Se incluyen estos elementos en la lista de h_1:

h_1: (8 2 3 4 5 6 7 1) **(2.2.5.3.14)**

- Al segundo hijo se le aplica un procedimiento análogo al anterior, cambiando los papeles de p_1 y p_2:

h_1: (8 4 5 2 6 7 3 1) **(2.2.5.3.15)**

Pueden encontrarse otros mecanismos de cruce aplicables a estas representaciones, como las debidas a Davis [Davis 85], a Syswerda [Davis 91, Capítulo 21] o a Fox [Fox 91].

2.2.6- Mecanismos de mutación

Estos operadores permiten que los individuos se desplacen con cierto grado de aleatoriedad de sus posiciones en el espacio de búsqueda. En aquellos procesos evolutivos en los que no existe el cruce, por ejemplo en las Estrategias Evolutivas, la mutación constituye un proceso fundamental, pues es el que lleva a cabo el movimiento de búsqueda, siempre guiada por los mecanismos de selección [Back 93].

En aquéllos algoritmos en los que sí está presente el cruce, el papel de la mutación pasa a un segundo plano en la evolución [Goldberg 89].(*)

Su papel es el de incorporar información genética no presente en los individuos iniciales. Además, en las últimas fases de la búsqueda, unos pocos individuos de buenas prestaciones pueden haberse difundido y acaparar gran parte de la población. El cruce entre individuos iguales no origina individuos nuevos, por lo que para continuar la búsqueda es necesario modificar a los individuos presentes. Es aquí donde la mutación cobra más importancia.

(*) Esta afirmación no está exenta de controversia (véase, por ejemplo [Murata 96]).

2.2.6.1- Mutación en representaciones binarias

La mutación se realiza de manera muy sencilla cuando los datos están codificados en cadenas de bits.

A cada uno de estos bits se le cambia de valor con una probabilidad fija e independiente entre unos a otros, como se aprecia en el ejemplo de la Figura 2.35.

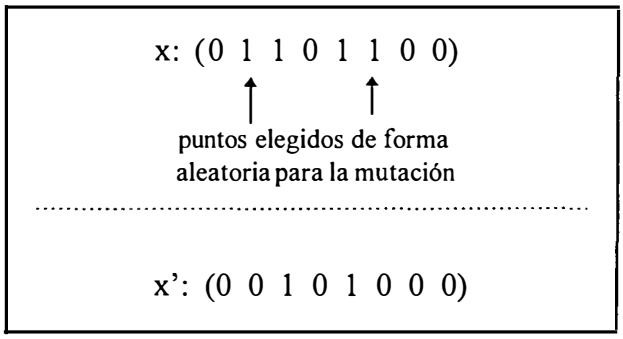

Figura 2.35 Mecanismo de la mutación

2.2.6.2- Mutación en representaciones reales

Cuando el individuo es un vector de J componentes

$$\bar{x} = (x_0, x_1, .., x_{J-1}) \qquad \textbf{(2.2.6.2.1)}$$

existen diversas alternativas para llevar a cabo la mutación. Sólo expondremos algunas de las más representativas.

a- Mutación Uniforme [Michalewicz 92]

Si cada una de las cuales toma valores en el intervalo

$$[x_{iL}, x_{iU}], \quad i = 0, 1, \ , J-1. \qquad \textbf{(2.2.6.2.2)}$$

Este operador se aplica con una probabilidad dada sobre cada individuo. Cuando uno de estos individuos es seleccionado para la mutación, todas sus componentes x_i tomarán un valores completamente aleatorios en los intervalos anteriores.

a- Mutación No Uniforme [Michalewicz 92]

Existirá aquí una variable binaria que se lanzará D veces, una por cada componente x_i del vector $\bar{x}$. A x_i se le asignará un valor uniformemente distribuido dentro del intervalo

$$\begin{array}{ll} [x_{iL}, x_i] \ , & \text{si la variable binaria tomó el valor 0} \\ [x_i, x_{iU}] \ , & \text{si la variable binaria tomó el valor 1} \end{array} \qquad \textbf{(2.2.6.2.3)}$$

Así el operador actúa de forma más local que en la mutación uniforme, impidiendo saltos demasiado bruscos en los individuos. A veces estos desplazamientos aleatorios son efectuados según leyes gaussianas, en vez de uniformes. Es el caso de las Estrategias Evolutivas [Schwefel 81] o de la Evolución Simulada [Fogel 91].

2.2.6.3- Mutación en listas de enteros

Volviendo a la representación de las listas como conjuntos fijos de elementos, sólo sujetos a reordenamientos, pueden encontrarse varios operadores adecuados.

a- Intercambio Recíproco [Herdy 90]

Dos elementos (ciudades) se intercambian de posición

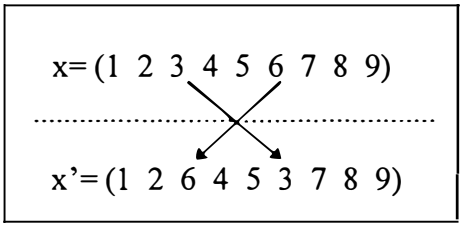

Fig. 2.36 Intercambio Recíproco en listas de enteros

b- Inserción [Herdy 90]

Se selecciona una ciudad y se le cambia de lugar de forma completamente aleatoria.

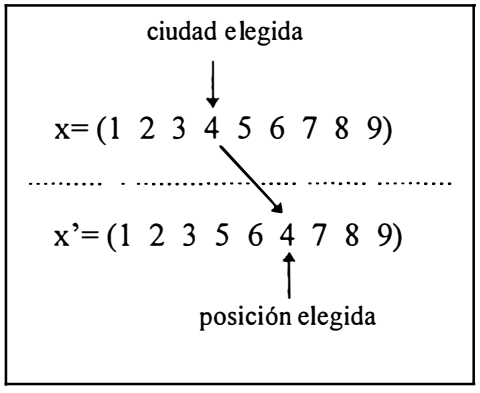

Fig. 2.37 Inserción en listas de enteros

c- Inversión [Herdy 90]

Se seleccionan dos puntos de la lista y a la sublista resultante se le da la vuelta:

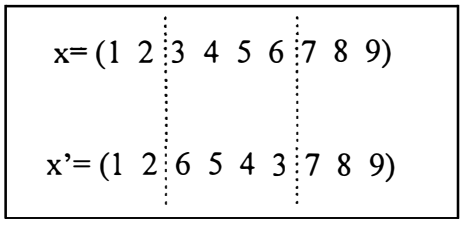

Fig. 2.38 Inversión en listas de enteros

d- Mutación Barajada [Davis 91]

Parecida a la anterior, sólo que en vez de invertir la sublista seleccionada, se permutan aleatoriamente sus componentes.

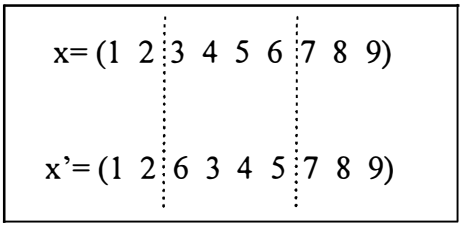

Fig. 2.38 Mutación Barajada

2.2.7- Otros operadores y mecanismos genéticos

Aparte de la mutación y el cruce se han diseñado multitud de operadores que intervienen en la generación de individuos nuevos de la población [ICGA 85], [ICGA 87], [ICGA 89], [ICGA 91], [ICGA 93], [Rawlins 91], [Buckles 92], [ICEC 96], [Goldberg 89], [Michalewicz 92], [Davis 91], etc. Su aplicabilidad varía desde los más generales hasta aquéllos específicos a la solución de un problema concreto.

A continuación comentamos algunos de ellos, de ámbito general.

2.2.7.1- Dominancia y diplocidad

En el núcleo de cada célula de un individuo (exceptuando aquéllas dedicadas a la reproducción sexual) existen varios cromosomas organizados en parejas. En cada pareja uno de los cromosomas procede del padre del individuo y otro de la madre. Estos cromosomas son homólogos en el sentido de que, por cada gen (componente genético) del cromosoma recibido del padre y referido a una característica específica(*), existe un gen en el cromosoma recibido de la madre y referido a esa misma característica.

Así existirá un gen para el color de los ojos en el cromosoma recibido del padre y otro, también para el color de los ojos, del recibido de la madre. Sin embargo, sólo uno de ellos será el que intervenga en las características físicas del hijo, dependiendo del valor que tomen ambos genes en el alfabeto AGCT ya comentado. A los distintos valores posibles del gen los llamaremos alelos. La expresión fenotípica de un gen dependerá, por tanto, sólo de la pareja de alelos existente y no de su procedencia paterna o materna. Así podemos pensar en el par cromosómico como en dos cadenas de genes enfrentadas como en la Figura 2.39.

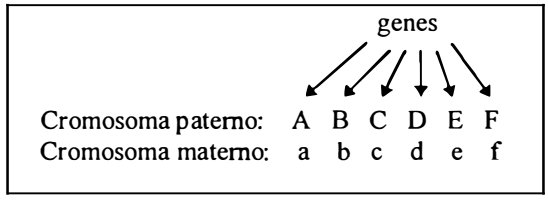

Fig. 2.39 Cromosomas diploides

(*) En realidad esto es sólo una simplificación: por un lado es difícil establecer con claridad la definición de gen y sus límites en la cadena cromosómica; por otro, no existe una relación biunívoca entre genes y características fenotípicas, siendo un hecho que un mismo gen puede influir en varias características del individuo, a la vez que una misma característica esté determinada por varios e incluso gran cantidad de genes [Dawkins 93].

La característica asociada al primer gen será la correspondiente al alelo 'A' o al 'a', según cuál de ellos sea dominante frente al otro.[2]

A esta duplicidad de la información genética se le da el nombre de diplocidad y a la prevalencia de unos alelos frente a otros, dominancia. Es importante señalar que la dominancia no es algo fijo, ni en el conjunto de individuos de una especie o de una población, ni en el transcurso de las generaciones. Por el contrario, no sólo es variable, sino que se encuentra asimismo sujeto a los propios vaivenes de la evolución genética.

La razón de ser de este "capricho" de la Naturaleza es, en palabras de Goldberg, "que la diplocidad proporciona un mecanismo para recordar alelos y combinaciones de alelos que fueron útiles previamente mientras que la dominancia proporciona un operador que protege a aquellos alelos recordados, de la ("dañina") selección en un medio hostil [Goldberg 89, pg. 149]. Como explica Goldberg más adelante, la Naturaleza, en el curso de su Historia, va cambiando sus condiciones, a menudo con brusquedad, y muchas veces de forma cíclica. Tiene sentido, por tanto, no destruir los planos que resultaron adecuados para la construcción de buenos individuas en un pasado en condiciones ambientales diferentes a las actuales, pero que pudieran volver a presentarse.

En el contexto de los AG, Hollstien y más tarde Holland [Holland 75] concibieron una representación de individuos expresada en dos cadenas homólogas de símbolos ternarios. En éstas el valor de cada símbolo (alelo) tiene el siguiente significado:

'0' → '0' en el contexto binario tradicional;
'1' → '1' recesivo: en el valor (fenotipo) del individuo se considerará un '1' binario si el gen homólogo presenta otro '1';
'2' → '1' dominante: el valor fenotípico del individuo corresponde a un '1' binario, cualquiera que sea el alelo homólogo.

De esta forma se establece el siguiente cuadro de dominancia (Figura 2.40) en el que los ejes cartesianos corresponden a los alelos de los dos padres y el valor en las casillas interiores indica cuál de ellos es el dominante.

	0	1	2
0	0	0	1
1	0	1	1
2	1	1	1

Figura 2.40 Cuadro de dominancia de los alelos de Hollstien

Como puede intuirse, este operador útil para la optimización de funciones no estacionarias, no supone ningún otro cambio en la estructura o funcionamiento del AG, el cual someterá a evolución, no sólo a los individuos de su población, sino también a la dominancia de sus genes.

[2] En algunos casos los fenotipos resultan de combinaciones intermedias de las expresiones fenotípicas de los alelos del padre y de la madre.

2.2.7.2- Nichos y especies

En el medio natural las distintas especies suelen estar muy especializadas en una actividad o mecanismo concreto conformado a lo largo de millones de años, que les ha conferido una ventaja diferencial frente a otros competidores y gracias a la cual han sobrevivido: es el caso de la fuerza en el león, el mimetismo en el insecto palo, la capacidad de vuelo en el albatros, la estructura social-familiar en las abejas o la inteligencia en el hombre. Estos pequeños espacios de privilegio en el gran orden natural reciben el nombre de nichos ecológicos y son ocupados por especies o razas dentro de las especies.

La evolución natural ha establecido en sus "empíricas deducciones" que el mejor mecanismo para que una especie ocupe uno de estos nichos es que sus individuos se apareen entre sí, evitando el cruce entre especies diferentes.(*)

Volviendo al más prosaico problema de la optimización, cuando la función que se quiere optimizar (pensemos en este apartado que se trata de maximización) es multimodal, como la que aparece en la Figura 2.41, los mecanismos de selección y pro-

Figura 2.41 Función multimodal con picos ocupados por individuos de un AG simple

creación conducen, con gran probabilidad, a explorar uno sólo o un conjunto reducido de sus picos (nichos). Existen dos inconvenientes a este comportamiento.

En primer lugar, no tiene mucho sentido que individuos bien adaptados en dos nichos distantes se crucen entre sí, pues, con gran probabilidad, se perderán sus características ventajosas de pertenencia a cada nicho. En el mecanismo propuesto en [Hollstein 71], al estilo de los Algoritmos Genéticos de Crianza (Breeding Genetic Algorithms) [Mulenbiem 93], sólo se permite el cruce de individuos próximos entre sí, con lo cual se consigue una exploración muy intensa de algunos nichos. Sin embargo, cuando los individuos que "colonizan" estos nichos (familias o etnias) dejan de mejorar sus prestaciones, se permite el cruce entre familias diferentes. El nombre que Hollstien dio a esta técnica es ciertamente explicativo *"inbreeding with intermitent crossbreeding"* (*"crianza intrafamiliar con crianza interfamiliar intermitente"*).

El segundo problema es que, en muchas aplicaciones, es interesante obtener soluciones en todos o al menos en buena parte de los picos de una función multimodal.

Goldberg propone modificar la función de coste y hacer que sea inversamente proporcional a una función que mida el número de individuos que ocupan un nicho y la proximidad entre ellos (a nivel genotípico o fenotípico). De esta forma se penalizan los

(*) Dawkins, argumentando estos potulados presenta un simpático ejemplo: "... como en el caso del cruce entre los caballos y los burros, el costo, por lo menos para la hembra, puede ser considerable .... sobre todo en el tiempo que podría haber invertido en criar a otros hijos. Luego, cuando la mula alcanza la edad adulta, resulta que es estéril. Las burras deberían ser muy muy cuidadosas en asegurarse que el individuo con el que copulan es otro burro y no un caballo". [Dawkins 93, pg. 212].

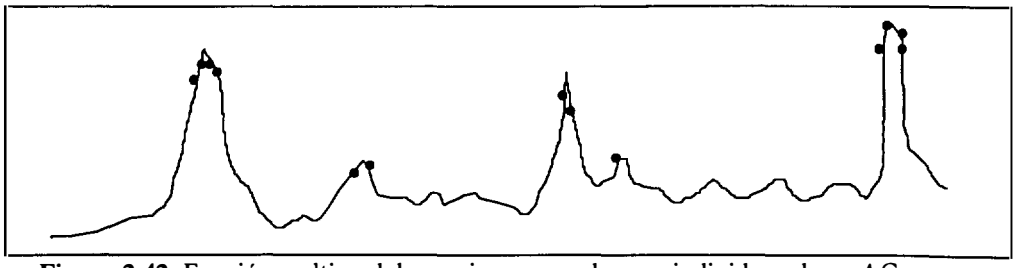

Figura 2.42 Función multimodal con picos ocupados por individuos de un AG que favorece la "especiación" (diversificación en la ocupación de estos picos).

nichos excesivamente populosos, favoreciendo un reparto más uniforme entre ellos, ajustado para que se tengan proporcionalmente más individuos cuanto mayor sea la bonanza de los nichos (amplitudes de los máximos).

2.2.7.3- Algoritmos Genéticos Basados en Virus

La Teoría de la Evolución basada en los Virus constituye una de las más modernas tendencias para explicar la Evolución Natural, tomando como fundamento la Biología Molecular [Anderson 70]. Según esta Teoría, la transmisión vírica es un mecanismo fundamental para transportar segmentos de DNA a través de las especies (propagación horizontal). Este material genético se transmitirá, a su vez, de generación en generación a través de los mecanismos evolutivos convencionales dentro de una especie (propagación vertical).

Los Algoritmos Genéticos Basados en Virus [Kubota 96] tratan de emular de forma simplificada este doble camino de propagación de la información codificada.

En ellos coexisten dos poblaciones, una de individuos normales codificados binariamente (cromosomas) y otra de virus, formados por subcadenas de los cromosomas anteriores. El traspaso de información genética, además de mediante la mutación y cruce convencionales, se realiza por medio de los virus, a través de dos operadores con funciones básicamente inversas:

- Transducción: el virus se forma como una copia de una parte de un cromosoma elegida al azar (Figura 2.43).

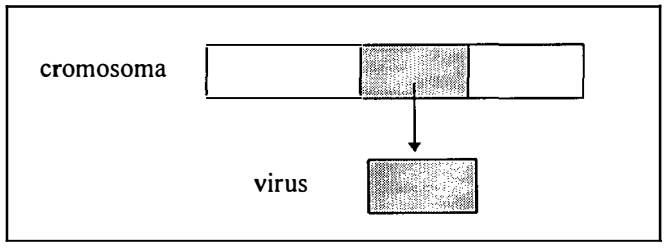

Fig. 2.43 Transducción

- Transcripción Inversa: el virus infecta a un cromosoma copiando sus bits en una subcadena de éste elegida al azar (Figura 2.44).

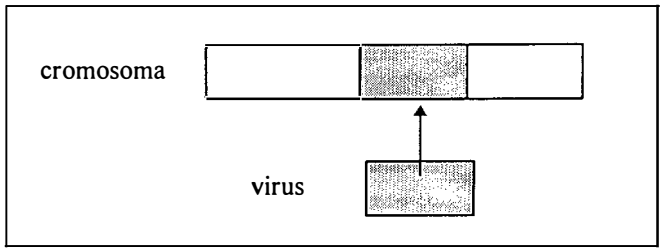

Fig. 2.44 Transcripción Inversa

El periodo de existencia de un virus es limitado y depende de la mejora (o empeoramiento) en la adecuación ocasionada en los individuos infectados. Con ello el virus también está sujeto a las leyes evolutivas.

2.2.8- Parámetros variables

Los valores óptimos que han de tomar los distintos parámetros de los algoritmos evolutivos (las probabilidades de cruce y de mutación, el número de individuos, etc.) son "a priori" difíciles de estimar, pues en general, depende en gran medida del problema concreto en el que éstos son aplicados. Algunas investigaciones [De Jong 75] [Grefenstette 86], [Schaffer 89] han estudiado el comportamiento de los AG sobre diversas funciones cuya optimización presenta elevada complejidad, intentando observar las implicaciones que los valores de los parámetros del AG tienen sobre su eficiencia. Así se han llegado a obtener juegos de parámetros óptimos para las funciones estudiadas, e incluso robustos ante otras funciones independientes de las anteriores, utilizadas para validar las conclusiones de los anteriores trabajos.

En cualquier caso, resulta difícil asumir que un juego de parámetros que resulta eficiente para la optimización de un conjunto de funciones, vaya a ser adecuado para optimizar funciones de cualquier tipo.

Por otra parte nada parece indicar (mucho menos los citados estudios) que la mejor opción sea mantener fijos los valores de estos parámetros a lo largo de las generaciones. Resulta, pues, conveniente que éstos puedan variar en la ejecución del algoritmo. Veamos varias maneras de lograrlo.

2.2.8.1- Parámetros con variación temporal establecida

a- En [Fogerty 89] se describe una aplicación de los AG en el campo del diseño industrial y, entre otras propuestas novedosas, se incluye una probabilidad de mutación que decrece de forma exponencial con las generaciones. Aunque en el citado trabajo no se acaba de explicar la lógica que inspira esta eleción, los resultados confirman su adecuación al problema resuelto.

b- Por su parte, Michalewicz dota a su operador de Mutación No Uniforme, comentado en el Apartado 2.2.6.2, de un comportamiento variable en el tiempo [Michalewicz 92]. A x_i se le dará un valor x_i' de acuerdo con:

$$x_i' = x_i + \Delta(t, x_i - x_{iL}) \quad , \quad \text{si la variable binaria tomó el valor 0} \qquad \textbf{(2.2.8.1.1.a)}$$

$$x_i' = x_i + \Delta(t, x_i - x_{iU}) \quad , \quad \text{si la variable binaria tomó el valor 1} \qquad \textbf{(2.2.8.1.1.b)}$$

donde Δ(t, y) retorna un valor aleatorio en el rango [0, y] tal que la probabilidad de que sea más próximo a 0 aumenta con la generación t. Así se permite explorar el espacio de búsqueda más uniformemente al principio y más localmente en etapas posteriores. Michalewicz propone la función

$$\Delta(t,y) = y \cdot \left(1 - r^{\left(1-\frac{t}{T}\right)^b} \right) \qquad \textbf{(2.2.8.1.2)}$$

siendo r y b parámetros de la función y T el número máximo de generaciones que el algoritmo vaya ejecutar.

2.2.8.2- Parámetros adaptativos

Consideramos aquí aquellos algoritmos cuyos parámetros genéticos varían de acuerdo con leyes prefijadas dependientes, de alguna manera, de las prestaciones de la población en una generación dada o de los individuos específicos sobre los que van a ser aplicados.

a- Davis idea una función de adecuación para los propios operadores genéticos que refleja las veces que cada uno de ellos ha sido responsable de la generación de buenos individuos en la población (individuos más adaptados que cualquiera de la generación anterior) y la magnitud de la mejora acarreada [Davis 89].

Estos operadores entran en juego con probabilidades acordes con las adecuaciones que presentan.

b- Por su parte, Patnaik hace que las probabilidades de cruce y de mutación sean dependientes de la función de adecuación promediada entre los individuos de la población y también de la función de adecuación de los propios individuos sobre los que se va a aplicar estos operadores [Patnaik 96]. Así, estas probabilidades van variando no sólo a lo largo de las generaciones, sino también de individuo a individuo. En realidad, las adecuaciones anteriores se miden con respecto a la adecuación máxima (la del individuo mejor dotado) como se observa en las leyes que rigen las probabilidades de cruce y mutación:

$$p_c = k_c \frac{f_{max} - f'}{f_{max} - \bar{f}} \quad \text{con} \quad 0 < k_c \leq 1 \qquad \textbf{(2.2.8.2.1)}$$

$$p_m = k_m \frac{f_{max} - f}{f_{max} - \bar{f}} \quad \text{con} \quad 0 < k_m \leq 1 \qquad \textbf{(2.2.8.2.2)}$$

donde f, $\bar{f}$, f_{max} y f' son, respectivamente, las función de adecuación del individuo sobre el que se va a aplicar el operador, la adecuación media de la población, la del mejor individuo dentro de la población y la de mejor individuo entre los dos sobre los que se va a llevar a cabo el cruce.

Lo que se pretende es, por una parte, introducir un sesgo en las alteraciones de tal forma que los individuos mejor adaptados sean modificados con menos probabilidad

que los peor adaptados, y, por otra parte, que estas probabilidades aumenten cuando el algoritmo no vea mejoradas globalmente sus soluciones (estancamiento).

c- Otra línea diferente es la de aquellos operadores cuyo funcionamiento se adapta al entorno, gobernado en algún aspecto por otro mecanismo evolutivo (metaoperador). Es el caso de las Estrategias Evolutivas [Schwefel 81], en las que los individuos tienen dos componentes:

$$(\bar{x},\bar{\sigma}) \qquad \textbf{(2.2.8.2.3)}$$

$\bar{x}$ indica un punto del espacio de búsqueda y $\bar{\sigma}$ es un parámetro que controla la magnitud posible de la mutación. La regla que determina el funcionamiento de la mutación es:

$$\bar{\sigma}' = \bar{\sigma} \cdot \exp(N(0,\Delta\sigma)) \qquad \textbf{(2.2.8.2.4)}$$

$$\bar{x}' = \bar{x} + N(0,\sigma') \qquad \textbf{(2.2.8.2.5)}$$

donde $(\bar{x}',\bar{\sigma}')$ es el individuo nuevo tras la mutación y $N(0,\Delta\sigma)$ una función que entrega una variable aleatoria gaussiana, de media nula y varianza $\Delta\sigma$.

Se puede apreciar que este operador también se ve afectado por el proceso de mutación genética. Por otra parte, los dos operadores de cruce definidos en las Estrategias Evolutivas también se aplican sobre la variable $\bar{x}$ y el parámetro $\bar{\sigma}$.

Otros operadores adaptativos pueden encontrarse en [Schaffer 87] , [Hansen 96] o [Smith 96].

2.2.9- Inclusión de restricciones

A veces los problemas de optimización se plantean sujetos a restricciones. Formalmente, se trata de encontrar $\bar{x}_{min} \in S$ tal que,

$$f(\bar{x}_{min}) \leq f(\bar{x}), \qquad \forall \bar{x} \in S \qquad \textbf{(2.2.9.1)}$$

y sujeto a:

$$\begin{aligned} g_i(\bar{x}) &= \bar{a}_i \qquad i = 1,2, \ldots ,N \\ h_j(\bar{x}) &\leq \bar{b}_j \qquad j = 1,2, \ldots ,M \end{aligned} \qquad \textbf{(2.2.9.2)}$$

siendo $g_i(\bar{x})$ y $h_j(\bar{x})$ funciones específicos del problema, y $\bar{a}_i$ y $\bar{b}_i$ constantes del mismo.

En estos casos algunas representaciones y mecanismos de alteración asociados a ellas pueden resultar muy inadecuadas para el progreso de un algoritmo de búsqueda hacia soluciones válidas de buenas prestaciones. Este es el caso del AG básico, que resulta demasiado encorsetado en sus representaciones binarias, con sus cruces y mutaciones.

Existen dos importantes técnicas que ayudan a los AG en su tarea de optimización.

a- Mediante la Penalización se construye una función de evaluación, suma de la función de adecuación más un término de penalización que castiga a los individuos que no cumplen las restricciones, tanto más cuanto más se separan de éstas:

$$e(x) = f(x) + P(x) \qquad \textbf{(2.2.9.3)}$$

donde e(x), f(x) y P(x) son las funciones de evaluación, de adecuación y de penalización, respectivamente.

La función de penalización puede, asimismo, depender de la generación en la que se encuentra el algoritmo [Richardson 89]; así puede conseguirse que al principio la búsqueda tenga mayor flexibilidad para investigar en puntos no permitidas del espacio, si bien, a medida que ésta avanza, se va exigiendo de forma más estricta el cumplimiento de las restricciones.

Este tipo de solución es, en principio, precario, dada la dificultad de ajustar equilibradamente la función de penalización: si su peso es pequeño, no será capaz de eliminar soluciones no válidas, y si es demasiado grande, hará que el proceso sólo se concentre en buscar soluciones válidas, sin favorecer suficientemente el sesgo hacia la mejor adecuación.

b- La Reparación consiste en mecanismos específicos mediante los cuales los individuos recién generados que se encuentren fuera de las regiones permitidas sean reformados en nuevos individuos que sí cumplen las restricciones.

Es evidente que estos operadores son necesariamene muy dependientes del problema y, en muchos casos, pueden conllevar una complejidad considerable que les puede hacer incluso inviables.

Según las consideraciones al respecto de ambos mecanismos y de acuerdo con algunos autores [Michalewicz 92], [Davis 91], lo más sensato es construir los algoritmos evolutivos ajustando sus componentes (estructuras de datos, operadores y parámetros) lo más adecuadamente posible al problema de optimización que se trata de resolver.

2.2.10- Incorporación de optimizadores locales

En gran parte de los problemas de optimización, sobre todos aquéllos que surgen de aplicaciones reales, es posible encontrar información específica, inferida deductivamente o basada en la experimentación, frecuentemente usada por los métodos más convencionales (si es que existen).

Si los algoritmos evolutivos no incorporan este tipo de conocimiento, se encontrarán en desventaja frente a los anteriores, precisando muchos más recursos computacionales para acercarse a las prestaciones de aquéllos, o muchas veces siendo incapaces de hacerlo. La generalidad de los algoritmos evolutivos está reñida con la especificidad requerida para dar soluciones a problemas concretos de forma eficiente. La manera natural de salvar esta contrariedad es incorporar en el proceso evolutivo la información específica del problema, aunque vaya en detrimento de su generalidad.

En general, aunque el uso de los algoritmos híbridos que surgen de la anterior unión, quede restringido a algunas aplicaciones muy concretas, gran parte de su estructura puede ser reutilizada en muchos otros problemas [Goldberg 89].

Muchas veces, lo que hemos llamado conocimiento específico del problema consiste en procedimientos locales de optimización. La manera de incorporar estos optimizadores locales es diversa y da lugar a un pequeño reajuste en la estructura básica de los métodos evolutivos de la Figura 2.25.

a- Híbridos secuenciales:

Goldberg los llama "procedimientos en diferido" ("batch approaches") [Goldberg 89]. En éstos, primero se ejecuta el algoritmo evolutivo; a continuación el optimizador local es aplicado a un pequeño porcentaje de las mejores soluciones (Figura 2.45). De esta forma el evolutivo explora el espacio encontrando una o unas pocas zonas interesantes, sobre las que posteriormente el optimizador local llevará una búsqueda más intensa y dirigida.

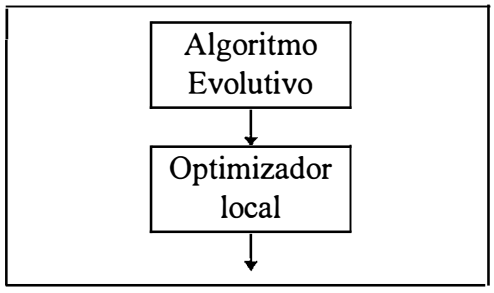

Fig.2.45 Híbrido secuencial

b- Estrategia de evolución Baldwiniana [Maley 96].

La estructura del proceso evolutivo es similar a la clásica, explicada en el apartado 2.2.2, con la diferencia de que se incluye el mecanismo de búsqueda local en el momento de efectuar la evaluación (Figura 2.46), sin modificar al individuo, que por

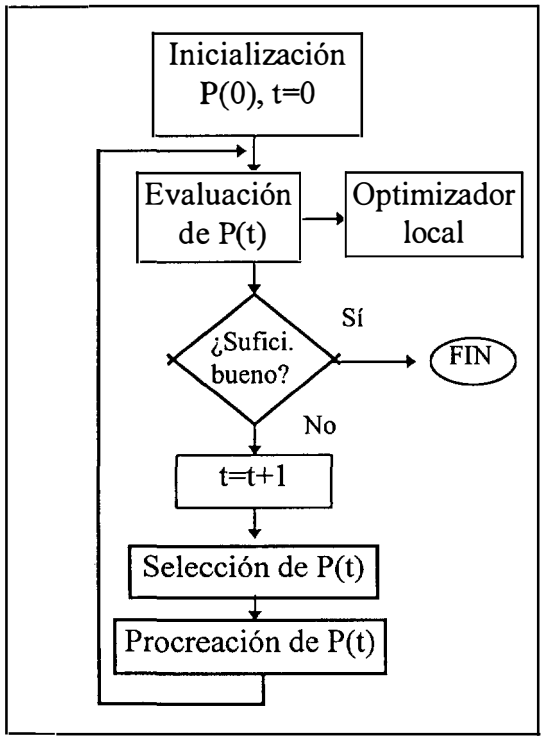

Fig. 2.46.a Estrategia Baldwiniana (diagrama de funcional).

Paso 1. Inicialización.

Paso 2. Evaluación de individuos (incluyendo la optimización local de los individuos).

Paso 3. SI se ha llegado a la convergencia o ya se han sucedido un cierto número de generaciones, parar.

Paso 4. Seleccionar los individuos más adaptados (padres).

Paso 5. Procrear nuevos individuos (hijos).

Paso 6. Volver al Paso 2.

Fig. 2.46.b Estrategia Baldwiniana (pseudocódigo).

tanto, no podrá transmitir las mejoras producidas por este mecanismo a las generaciones posteriores. La Naturaleza también ha adoptado este camino para evolucionar.(*)

c- Estrategia de evolución Lamarckiana [Maley 96].

La diferencia con el anterior es que el operador de aprendizaje local se incluye como otro operador más de alteración, que se aplica a cada individuo después de la selección. Con ello el "conocimiento adquirido" en este aprendizaje es pasado a través de las generaciones (Figura 2.47).

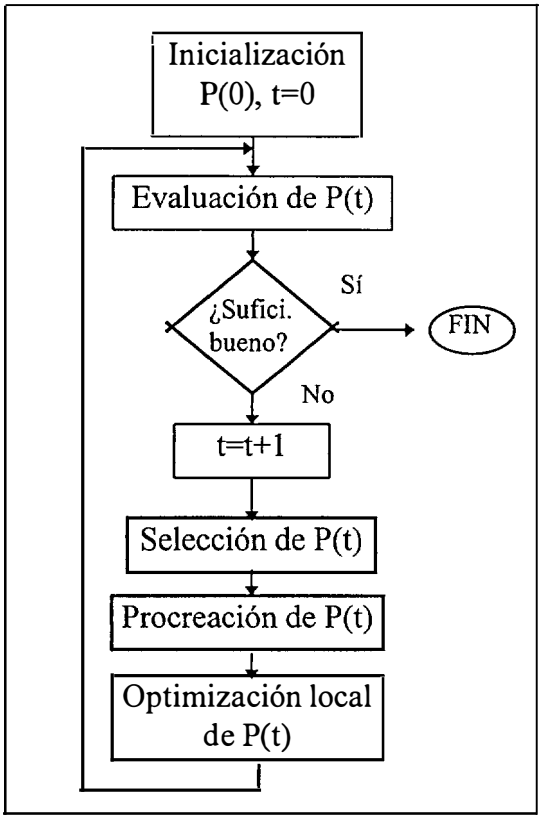

Fig. 2.47.a Estrategia Lamarckiana (diagrama de funcional).

Paso 1. Inicialización.
Paso 2. Evaluación de individuos (libros de código tentativos).

Paso 3. SI se ha llegado a la convergencia o ya se han sucedido un cierto número de generaciones, parar.

Paso 4. Seleccionar los individuos más adaptados (padres).

Paso 5. Procrear nuevos individuos (hijos).

Paso 6. Optizarlos localmente

Paso 7. Volver al Paso 2.

Fig. 2.47.b Estrategia Lamarckiana (pseudocódigo).

2.2.11- Eliminación de individuos repetidos

Si dos o más individuos idénticos (clones) aparecen en la población estarán recortando el contenido potencial de información de la población, el cual sería más rico si cada uno de ellos ocupara una posición diferente en el espacio de búsqueda.

Por otra parte, el riesgo de que varios de estos individuos "inunden" la población puede ser elevado, al menos cuando se usan representaciones discretas como en los AG, o cuando se incluyen en el algoritmo optimizadores locales que conducen al mismo

(*) Como argumenta Dawkins en cierto tono dramático: " .. las características adquiridas no son hereditarias. No importa cuántos conocimientos y cuánta sabiduría se adquieren durante una vida, nada pasará a los hijos por medios genéticos. Cada nueva generación empieza desde el principio" [Dawkins 93, pg.30].

punto a dos individuos próximos entre sí en el dominio de influencia de un atractor (mínimo o máximo) local (Figura 2.48).(*)

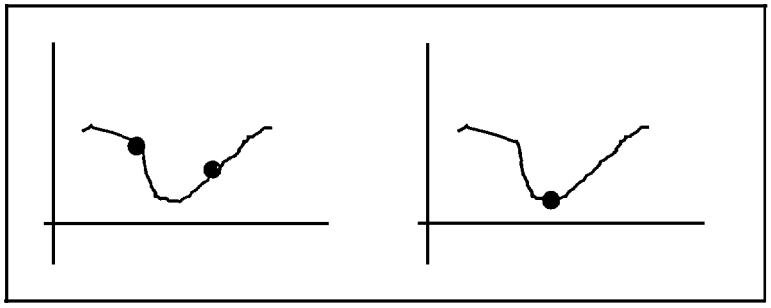

Fig. 2.48 Efecto de la optmización sobre dos puntos situados en el dominio de influencia de un atractor local.

Se pueden llegar a generar así conjuntos extensos de clones que impiden una búsqueda eficiente. Existen varias maneras de atajar este problema.

a- Remoción de Clones:

Si, tras la procreación, dos o más individuos resultan ser clones entre sí, uno de ellos será sometido a mutación hasta que ambos difieran de forma apreciable.(*)

b- Prevención de Incestos [Eshelman 91]:

Cuando se elige para el cruce una pareja de individuos idénticos, se rechazan y se elige otra.

Esta propuesta está diseñada para operar sobre un AG, donde los mecanismos de cruce más habituales, como los explicados en el Apartado 2.2.5.1, aplicados sobre individuos idénticos, dan lugar a hijos clónicos de sus padres.

En el caso de representaciones en punto flotante, algunos operadores de cruce, también darán origen a individuos clónicos, como ocurre con los dos operadores expuestos en el Apartado 2.2.5.2. Otras veces, el hecho de incluir cierta aleatoriedad, hará posible la obtención de hijos diferentes [Zhang 93], [Malanda 96].

Merece la pena apuntar aquí que también en el mundo natural el incesto puede tener consecuencias genéticas dañinas debido a que "los genes recesivos letales o semiletales surgen a la superficie" [Dawkins 93, pg.213].

2.2.12- Elitismo

En las técnicas evolutivas, el elitismo significa la inclusión en una generación nueva del mejor individuo de la generación anterior. En sus experiencias, De Jong encontró que en las superficies unimodales el elitismo proporciona muy buenos

(*) De todas formas, en el caso de representaciones en punto flotante, el criterio de igualdad entre dos individuos es demasiado exigente y, por tanto, inoperante. Es mejor relajar el concepto y pensar que dos clones son dos individuos que distan entre sí una distancia pequeña, en general, dependiente de las condiciones que enmarcan el problema, aunque no necesariamente nula.

(*) Al autor de esta Tesis no le consta la publicación de este operador genético en las referencias por él consultadas, sin embargo, parece el más sencillo y natural según el planteamiento anterior del problema.

resultados, pero su comportamiento se degrada cuando superficies de optimización presentan varios óptimos locales [De Jong 75]. De este hecho concluye en relacionar al elitismo con la búsqueda local, en detrimento de una perspectiva global.

Por otra parte, sin ninguna medida adicional, este operador puede conducir a la aparición de clones con los problemas comentados en el apartado anterior.

A pesar de estas circunstancias, para algunos autores [Schaffer 91], [Eshelman 91], el elitismo presenta interesantes ventajas, siendo utilizado con frecuencia en aplicaciones reales [Liepins 91], [Murthy 96], [Ishibuchi 96].

Por otra parte, este mecanismo se encuentra presente en las Estrategias Evolutivas, en las que el proceso de selección es completamente determinístico [Back 93]. En la variante $ES(\mu,\lambda)$, de una población de μ individuos se generan λ hijos ($\lambda>\mu$), de los cuales se eligen los μ mejores. En la variante $ES(\mu + \lambda)$, de μ padres se generan λ hijos y se eligen los λ mejores entre la población conjunta de padres e hijos.

Capítulo 3

ANTECEDENTES

3.1- Introducción

En la que se puede considerar como la primera fase de su desarrollo, la Cuantificación Vectorial fue sólo considerada como una estrategia de codificación de fuente, en la que el efecto del canal no era tomado en cuenta. La tarea de encontrar libros de código óptimos para representar vectores de datos en estas condiciones "ideales" fue elegantemente tratada por Yoseph Linde y sus colegas Andrés Buzo y Robert M. Gray [Linde 80]. Su famoso algoritmo, conocido por la comunidad científica como LBG, consiste en la alternancia iterativa de la Regla del Vecino Más Próximo y la Regla de los Centroides, por medio de lo cual un libro de códigos dado se desplaza en el espacio de libros de código hacia un punto en el que la distorsión presenta un mínimo local. Debido a este carácter local, hacen falta muchas repeticiones del algoritmo con diferentes puntos de partida (libros de código distintos) para obtener una buena solución global. Varias aproximaciones ayudan al LBG a obtener un "buen" libro de códigos inicial [Liu 87], [Katsavouridis 94]. Por otro lado, en [Lee 97] se presenta una ligera modificación de las reglas básicas anteriores, basadas en el gradiente, que incluye un parámetro de tamaño de paso, similar al que existe en los algoritmos de filtrado adaptativo.

No obstante, la naturaleza multipolar del problema clamaba por técnicas de optimización más globales, que pudieron encontrarse en el Temple Simulado [Cetin 88], la Relajación Estocástica [Zeger 92], pariente cercano del anterior, los Algoritmos Genéticos [Choi 96] o las Redes Neuronales Autoorganizativas [Yair 92].

Según se explicaba en el apartado introductorio, la implantación en los años 90 de los sistemas de telefonía móvil, y la expansión de la TV digital y de los servicios y aplicaciones multimedia han aumentado sin cesar la importancia de la codificación y demandado al mismo tiempo soluciones robustas que permitieran calidades aceptables con canales sometidos a condiciones elevadas de ruido.

La Cuantificación Vectorial, ha corregido su rumbo apuntando hacia el diseño de librerías de código que resuelvan a la vez el problema conjunto de la compresión eficiente y la inmunidad al ruido de canal.

La primera de las tendencias ha consistido en buscar primero un buen libro de códigos, diseñado para la transmisión sin error de canal por alguno de los métodos existentes, y después encontrar una asignación de índices apropiada para usar el codificador en un entorno ruidoso, atendiendo a la siguiente directriz: debe asignarse a dos vectores próximos, dos índices binarios con pocos bits diferentes, de tal forma que la probabilidad de que un vector sea confundido (debido al ruido de canal) por otro cercano a él sea mayor que la probabilidad de que sea confundido por otro más alejado. Dentro de esta línea, pueden citarse soluciones "ad-hoc" como el Código Pseudo-Gray

de Zeger [Zeger 90] o el Algoritmo de Asignación de Indices de Wu [Wu 93], que coexisten con aproximaciones de mayor demanda computacional como las que hacen uso del Temple Simulado [Farvardin 90] o de Algoritmos Genéticos [Pan 96]. Sin embargo, esta estrategia no toma en consideración el hecho de que el diseño del libro de códigos y de la asignación de índices están interrelacionados y no debería hacerse de forma separada.

En una segunda línea, se han desarrollado algoritmos que optimizan la librería de códigos teniendo en cuenta las probabilidades de error de canal, sin dividir el problema en sus dos partes constituyentes. Este es el caso del Algoritmo Generalizado de Lloyd (explicado en el apartado 2.1.4.5) [Kumazawa 84] o el Enfriamiento Determinístico de Miller [Miller 94].

Otros algoritmos han enfocado el doble problema de una forma explícita basada en la repetición del siguiente bucle: un paso para optimizar la librería de códigos, seguida de otro paso para optimizar la asignación de índices. La aproximación basada en Decisión Blanda de Cuperman [Cuperman 94], dentro de la estirpe de algoritmos de Enfriamiento Simulado es un ejemplo de esta vía.

En los siguientes apartados de este capítulo se revisan todas estas tendencias. En una primera parte, se repasan los métodos de diseño de CV, sin consideración de errores en el canal; por un lado, aquéllos cuya optimización es de carácter local y que no son sino variaciones del LBG convencional, y por otro, los que plantean una búsqueda de los libros de código basada en técnicas heurísticas más generales: métodos termoestadísticos, genéticos, neuronales o "fuzzy", como luego comentaremos.

La segunda mitad del capítulo se centra en la descripción de los métodos de CV en los que sí se consideran los errores que introduce el canal. Esta parte, a su vez, se desglosa en las tres tendencias comentadas: métodos de reordenamiento de índices, métodos que optimizan conjuntamente vectores e índices, y métodos que lo hacen de forma iterativa y alternada.

3.2- Optimización de CV sin errores de canal

3.2.1- Optimización local

La gran mayoría de los métodos de diseño de CV están relacionados en mayor o menor medida con las Reglas del Vecino Más Próximo y de los Centroides, y, por consiguiente, con el algoritmo LBG.

En este apartado se recogen aquéllos que pueden considerarse variaciones de este algoritmo, o bien en la inicialización de la librería de vectores, o bien, en las reglas de optimalidad. La naturaleza de la búsqueda que todas ellas efectúan es local, herencia del algoritmo LBG.

3.2.1.1- Mejoras en la inicialización del algoritmo LBG

Existen varias técnicas que mejoran el procedimiento básico de inicialización del algoritmo LBG, consistente en la elección aleatoria de L vectores dentro del conjunto de entrenamiento [Gersho 92]. Comentaremos dos de ellas, no citadas en esta referencia.

a- Procedimiento de inicialización de Liu [Liu 87].

Trata de reducir al máximo la aleatoriedad en la confección del libro de códigos inicial. En este procedimiento existe un parámetro que controla el proceso de inicialización, δ que define la máxima distorsión tolerable entre dos subparticiones o celdas.

El primer elemento del libro de códigos se selecciona aleatoriamente entre el conjunto de vectores de entrenamiento. El resto de estos vectores se incluye en alguna celda de las existentes si su distancia al centroide de esa celda es menor que δ. Si no encuentra ninguna celda donde poder incluirse, forma su propia celda.

Si el número de celdas resultantes M es mayor que el que se desea (L), se disminuye δ y se repite el proceso anterior tantas veces como se precise, hasta que $M \geq L$. Si $M > L$ se combinan las celdas de manera apropiada, de acuerdo con algún criterio basado en su distorsión o su población, para que al final queden exactamente L de ellas.

b- La propuesta de Katsavounidis [Katsavounidis 94].

Está basada en buscar vectores del conjunto de entrenamiento que estén lo más alejados entre sí posible. El procedimiento es el siguiente: como primer vector código se elige el de mayor norma euclídea de todo el conjunto de entrenamiento. El segundo vector código será el vector de entrenamiento más alejado del primer vector código; el tercero, el más alejado del segundo, y así sucesivamente hasta completar los L vectores código de la librería.

Como puede observarse esta manera de diseñar el libro de códigos inicial carece por completo de aleatoriedad.

3.2.1.2- Modificaciones al algoritmo LBG

a- División ("Splitting")

En realidad esta propuesta aparece en el mismo artículo en el que se da a la luz el propio algoritmo LBG [Linde 80], presentándose como una variante de éste.

El método de División va obteniendo libros de códigos de tamaño creciente (cada uno doble del anterior) de la siguiente manera:

- el primer libro de códigos, de tamaño 1, es el centroide de todo el conjunto de entrenamiento;
- este centroide se divide en dos muy próximos entre sí:

$$\bar{c}_i \begin{cases} \nearrow \bar{c}_i + \bar{\varepsilon} \\ \searrow \bar{c}_i - \bar{\varepsilon} \end{cases} \qquad \textbf{(3.2.1.2.1)}$$

siendo $\bar{\varepsilon}$ una pequeña perturbación fija.

- se aplica el algoritmo LBG a esta librería de dos vectores
- se repite el proceso anterior de división y relocalización (LBG) hasta que el tamaño de la librería sea el deseado L.

b- LBG acelerado [Wu 94]

Aquí se presenta una modificación en el cálculo del Vecino Más Próximo basada en consideraciones geométricas, con lo que se consigue acelerar considerablemente el algoritmo LBG, cuya componente más costosa es precisamente esta regla.

Como se ve en la Figura 3.1 y, atendiendo a la desigualdad triangular, el vecino más próximo al vector de entrenamiento $\bar{v}$ sólo hará falta buscarlo en el conjunto

$$B_i(\bar{v}) = \left\{ \bar{c}_j \mid \left\| \bar{c}_i - \bar{c}_j \right\|_2 < 2\left\| \bar{c}_i - \bar{v} \right\|_2 \right\} \qquad \textbf{(3.2.1.2.2)}$$

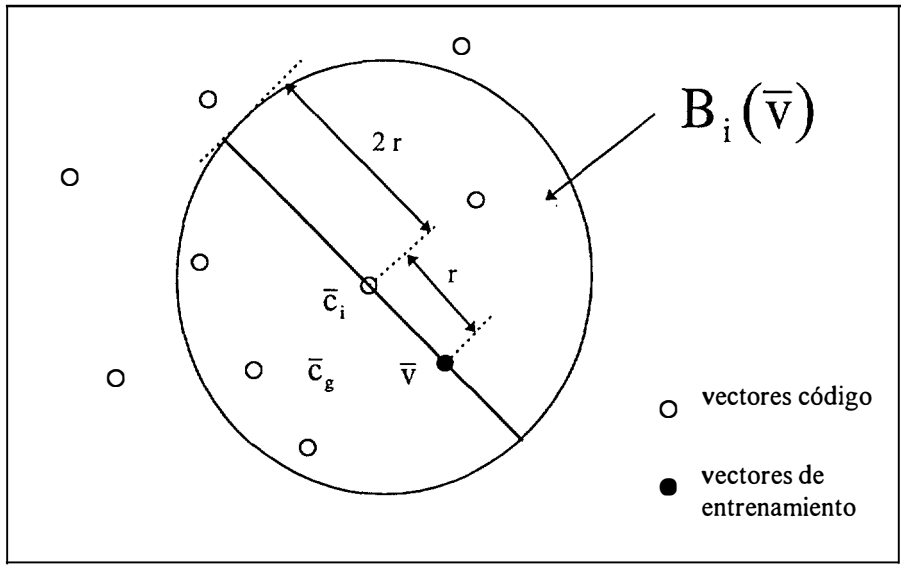

Fig. 3.1

pues todos los puntos externos a él distarán más de $\bar{v}$ que el vector $\bar{c}_i$.

El proceso que Wu propone es comenzar por un $\bar{c}_i$ cualquiera, en principio, con el que quedan excluidos todos los vectores código fuera de $B_i(\bar{v})$. Más tarde se selecciona otro vector de $B_i(\bar{v})$, por ejemplo el $\bar{c}_g$. Los centroides válidos ahora serán aquéllos incluidos en la intersección $B_i(\bar{v}) \cap B_g(\bar{v})$. De esta forma se avanza hasta que la última intersección sólo esté formada por un punto que será el vecino más próximo a $\bar{v}$.

Además, como en el proceso LBG, los vectores código y las particiones se suelen desplazar poco en cada iteración básica, se sugiere que cada vez que se vaya a buscar al vector código más próximo a un vector de entrenamiento, se comience la búsqueda a partir del vector código asociado a la partición en la que el vector de entrenamiento se encontraba en la anterior iteración.

c- Algoritmo de K-medias optimizado [Lee 97]

En realidad K-medias es el nombre que se le da en Estadística al algoritmo LBG [Anderberg 73]. Así este algoritmo se presenta como una sencilla modificación al LBG, en concreto a la Regla de los Centroides, según la cual el vector código no coincidirá exactamente con los centroides.

En una iteración t del algoritmo, el vector código $\bar{c}_i(t)$ se calcula a partir del centroide $\hat{c}_i(t)$ y del vector código de la iteración anterior. La ley seguida es:

$$\bar{c}_i(t) = \bar{c}_i(t \quad 1) + \mu(\hat{c}_i(t) - \bar{c}_i(t-1)) \qquad \textbf{(3.2.1.2.3)}$$

Si μ toma valor 1, esta ley coincidirá con la de los Centroides, pero si $\mu > 1$, el algoritmo da un paso mayor en esa misma dirección, con lo que se logra acelerar la convergencia del proceso. En cualquier caso, μ no debe superar nunca el valor de 2, pues podría hacer inestable el método.

3.2.2- Optimización global

Siendo el diseño de librerías de códigos un problema de optimización de alta complejidad, en el que los métodos locales como el LBG o las modificaciones explicadas en el apartado anterior no permiten explorar eficientemente el espacio en busca de mejores soluciones, muchos métodos de optimización global de carácter general han sido probados y, en muchos casos, incorporado con éxito al bagaje de herramientas de diseño de CV. Veremos en esta sección algunos de ellos.

3.2.2.1- Métodos termoestadísticos

a- Temple Simulado (TS)

Este procedimiento es muy utilizado en problemas complejos de optimización en los que las soluciones analíticas son infrecuentes y la existencia de varios, a veces muchos, óptimos locales hacen que las técnicas iterativas del tipo del gradiente no sean adecuadas para hallar el óptimo global.

El TS opera de la siguiente manera:

Sea T_0, T_1, T_2, ... una secuencia de números reales llamada esquema de temperaturas, donde

$$T_o \geq T_1 \geq T_2 \geq \quad \geq T_N \qquad \text{con} \lim_{N\to\infty} T_N = 0 \qquad \textbf{(3.2.2.1.1)}$$

Sea $f(\bar{x})$ la función que se trata de minimizar. A la temperatura T_n se calcula enlcoste correspondiente al vector $\bar{x}$. Seguidamente este vector se perturba aleatoriamente. Si el vector resultante presenta menos energía que el que le precedía, se acepta, de lo contrario se acepta sólo con una probabilidad dada por:

$$\exp\left(\frac{-\beta\,\Delta E}{T_n}\right) \qquad \textbf{(3.2.2.1.2)}$$

donde ΔE es el incremento de la energía debida a la perturbación y β, una constante del método. El procedimiento se continúa hasta que el número total de aceptaciones o rechazos exceda un cierto valor.

Con el esquema de temperaturas

$$T_n = \frac{C}{\log(n+1)} \qquad \textbf{(3.2.2.1.3)}$$

se asegura la convergencia al mínimo global, aunque no se indica el número de iteraciones necesarias [Zeger 92]. Un subóptimo algoritmo debido a Kirkpatrick resulta más rápido y cómodo:

$$T_n = T_0 \cdot A^n \qquad \text{con} \quad 0 \leq A \leq 1 \qquad \textbf{(3.2.2.1.4)}$$

El algoritmo TS puede aplicarse de forma directa a la CV [Cetin 88], haciendo que $\bar{x}$ sea el libro de códigos y f la distorsión media dada en (2.1.3.4.4). Los vectores del libro de códigos irán variando, haciendo decrementar la distorsión media hasta llegar a una situación de equilibrio. Sin embargo, si no se dota a este proceso de ningún otro mecanismo de búsqueda local, tardará muchas iteraciones en converger y lo hará, con gran probabilidad, a regiones subóptimas.

Una posible solución a este problema es la combinación de los algoritmos LBG y TS [Zeger 92].

En la Figura 3.2 se muestra el diagrama funcional y el pseudocódigo del algoritmo de TS propuesto en [Zeger 92] para el diseño de librerías de código en CV.

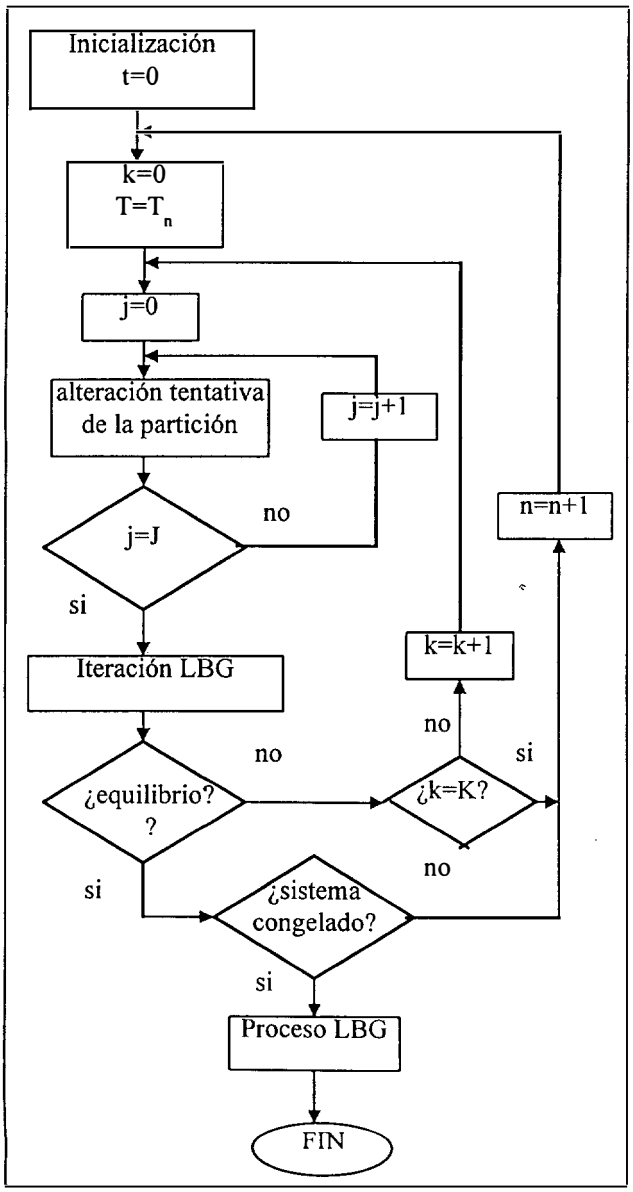

Fig. 3.2.a Temple Simulado (diagrama de flujo)

Paso 1. Inicialización.

Paso 2. Descenso de temperatura ($T=T_n$).
Inicializar contador (k=0).

Paso 3. Repetir J veces:
alteración tentativa de la partición.

Paso 4. Iteración LBG.

Paso 7. SI hay equilibrio
6.1- SI se ha alcanzado la congelación
Proceso LBG final
PARAR
6.2- SI NO
Volver al Paso 2.

Paso 7. SI NO hay equilibrio
7.1- SI k= 10
Volver a l Paso 2.
7.2- SI NO
Volver al Paso 3.

Fig. 3.2.b Temple Simulado (pseudocódigo)

La perturbación se implementa trasladando aleatoriamente la asignación de un vector de entrenamiento de una subpartición a otra cercana. Después se observa la variación en la distorsión global; si ésta disminuye se acepta el cambio, si no la aceptación se producirá con una probabilidad P dada por:

$$P = \exp\left(-\frac{\Delta E_n}{T_n}\right) \qquad \textbf{(3.2.2.1.5)}$$

esto es, menor cuanto más baja es la temperatura T_n y cuanto mayor es el aumento de distorsión ΔE_n.

Esta perturbación tentativa se llevará a cabo J veces, al final de las cuales se aplica una iteración LBG. Todo este proceso se repite sin variar la temperatura hasta que el descenso relativo de distorsión sea inferior a ε_1 desde la iteración LBG anterior, con un límite de K repeticiones.

En este punto se hace descender la temperatura según la ley (3.2.2.1.4). Cuando en una temperatura dada, sólo la porción ε_2 de las perturbaciones tentativas del codificador son aceptadas, se declara el sistema congelado y se termina el proceso lanzándose el GLA hasta que la distorsión converja.

Otra alternativa en vez de perturbar las subparticiones y aplicar a continuación la Regla de los Centroides, es modificar de forma directa los vectores código, y luego hacer uso de la Regla del Vecino Más Próximo. En este caso, el nuevo cálculo de la distorsión sería mucho más costoso, pues tras cada perturbación habría que recalcular todas las subparticiones.

b- Relajación Estocástica [Zeger 92]

Pariente cercano al TS, en este algoritmo todas las perturbaciones son aceptadas y seguidas de una iteración del algoritmo LBG. Más en concreto, se perturba todo el conjunto de entrenamiento, añadiendo a cada vector un ruido gaussiano de media nula y varianza decreciente con la temperatura del proceso. Tras esto se calculan las subparticiones según el Vecino Más Próximo. Los centroides se hallan a partir de los vectores de entrenamiento asociados a cada subpartición, pero sin considerar el ruido añadido.(*)

El esquema de temperaturas, y con éste la varianza del ruido, sigue una ley decreciente que puede ser:

$$T_n = \sigma_x^2(n) = \sigma_x^2(0)\cdot\left(1+\frac{n}{I}\right)^p \qquad \textbf{(3.2.2.1.6)}$$

donde I es el número de iteraciones del proceso completo y p una constante empírica.

Este procedimiento es mucho más rápido que el TS, por cuanto no tienen que recalcularse la distorsiones para validar las perturbaciones. También la RE permite perturbar los centroides, en vez de los vectores de entrenamiento, aplicando luego la Regla del VMP.

(*) En contraste con Linde et. al. que presentan, también en [Linde 80], un esquema similar, en el que se hallan los nuevos centroides a partir de los vectores de entrenamiento con ruido añadido.

3.2.2.2- Métodos genéticos

a- Algoritmo Genético de Lloyd Generalizado [Choi 96]

Es un AG en el que los individuos son librerías de códigos tentativas y en el que la función objetivo es la relación señal distorsión dada por:

$$RSD = 10\log_{10}\left(\frac{D}{S}\right) \qquad \textbf{(3.2.2.2.1)}$$

donde D es la distorsión de cuantificación dada en (2.1.3.4.4) y S la potencia media de la señal.

Por otra parte se incluyen operadores de mutación y cruce en punto flotante convencionales, aunque ligeramente distintos a los expuestos en el apartado 2.2.5.2 y se utiliza una estrategia de selección consistente en reemplazar de forma directa (determinística) individuos de buenas prestaciones por individuos de malas prestaciones.

b- En [Kim 99] se hace uso de otro AG para el diseño de libros de código, aunque en este caso, el contexto es el de la CV constreñida, en concreto, una versión híbrida de CV multietapa y forma-ganacia. Las más reseñables características son:

- representación entera de los vectores código. Aquí la estructura de los datos es mucho más compleja que en el caso anterior, pues hay que considerar la forma y la ganancia de las diversas etapas;
- inicialización "quasi-determinística": los vectores código iniciales se toman a partir de los niveles correspondientes a la cuantificación uniforme, desplazados una cierta cantidad aleatoria.
- función de adecuación (fitness):

$$F = \frac{1}{1+D} \qquad \textbf{(3.2.2.2.2)}$$

donde D es la distorsión media dada en (2.1.3.4.4);

- reproducción por torneo;
- cruce: específico a la naturaleza dicotómica (forma-ganacia) de los datos, buscando además emparejar vectores código lo más similares posible.
- mutación: muy parecido a la uniforme
- elitismo;

3.2.2.3- Métodos neuronales

a- Mapas Autoorganizativos de Características [Kohonen 89]

En terminología inglesa reciben el nombre de SOFM (Self-Organizing Feature Maps) o simplemente SOM. Son Redes Neuronales (RN) que implementan proyecciones no lineales de espacios de dimensión grande (dominio de la señal) a espacios de baja dimensión (mapas o dominio de representación), por ejemplo arrays de 1, 2 ó 3 dimensiones, árboles, etc.

La proyección intenta preservar las relaciones topológicas entre los dos dominios, de tal forma que conjuntos de puntos próximos en el dominio de la señal se reflejen en puntos próximos en el dominio de representación.

Los elementos básicos de los Mapas Autoorganizativos son las neuronas, que se encuentran interconecatadas entre sí formando el dominio de representación y almacenan en su interior un vector perteneciente al dominio de la señal. Generalmente las componentes de este vector son referidos como los pesos de la neurona y a la propia neurona se le asocia de forma directa con el vector que alberga (así hablaremos de la localización de la neurona, refiriéndonos a la de localización de este vector), sin perder de vista la ligazón entre neuronas en el mapa de representación.

Al igual que en otras RN, el aprendizaje de las neuronas, se realiza a través de ejemplos: una a una van apareciendo muestras de la señal y las neuronas van desplazándose de acuerdo con alguna ley establecida "a priori", adquiriendo paso a paso su estructura organizada.

Para el caso de la CV, los vectores asociados a cada neurona formarán el libro de códigos. Se quiere, por tanto, representar un espacio de D dimensiones con un conjunto de L neuronas, dispuestas en filas o en el plano (o incluso en un espacio de dimensión reducida). Como veremos, estas redes hacen uso del aprendizaje competitivo. Cuando una de las muestras de entrenamiento llega a la red de neuronas, se observa cuál de todas ellas presenta un vector de pesos más próximo a ella (generalmente según una norma cuadrática). Esta neurona y aquéllas contenidas en su vecindad son modificadas, haciendo que sus pesos se acerquen a la muestra de entrada. La ley que regula el reajuste de las neuronas es [Kangas 90]:

$$\bar{c}_i(t+1) = \begin{cases} \bar{c}_i(t) + \alpha(t)\left(\bar{v}(t) - \bar{c}_i(t)\right), & \text{si} \quad i \in N_j(t) \\ \bar{c}_i(t), & \text{si} \quad i \notin N_j(t) \end{cases} \qquad \textbf{(3.2.2.3.1)}$$

donde $\alpha(t)$ es un parámetro escalar que hace las veces de paso de adaptación, decreciendo monótonamente con t, y $N_j(t)$ es la función de vecindad en torno a la neurona $\bar{c}_j$, la más próxima al vector $\bar{v}(t)$, que indica cuáles son sus neuronas vecinas en el mapa. Esta función va decreciendo con t, y con ella, el área de influencia de cada muestra presentada en el entrenamiento, haciendo que el aprendizaje sea gradualmente más local.

En realidad, bajo ciertas aproximaciones puede considerase a la ley anterior como una aproximación estocástica e iterativa del tipo gradiente a la minimización de la función de distorsión dada en (2.1.3.1.8) [Kohonen 96], [Pal 93], o bien a su versión para conjuntos finitos de datos dada en (2.1.3.4.4).

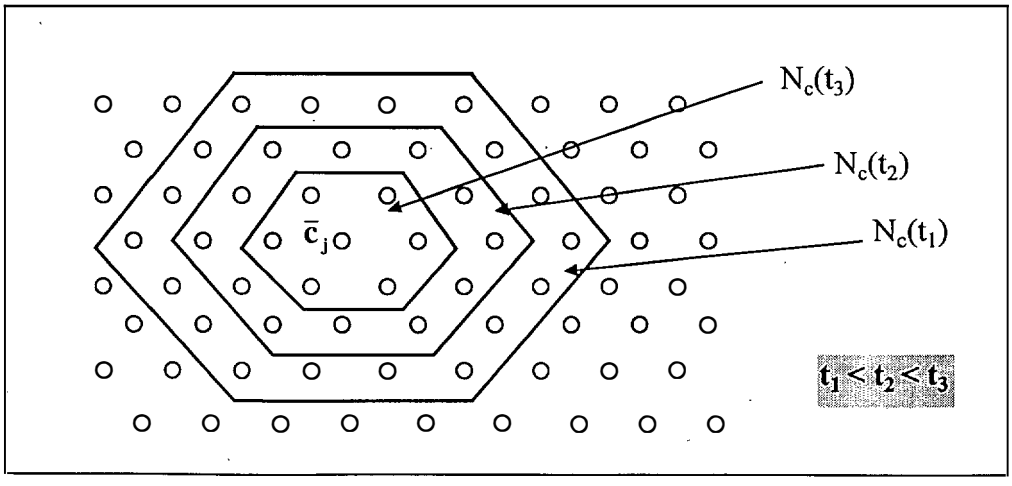

Fig. 3.3 Funciones de vecindad en un Mapa Autoorganizativo

El algoritmo que hace evolucionar los pesos de las neuronas y, por tanto los libros de código es el que se dibuja en la Figura 3.4. El equilibrio se asume después de haber pasado todo el conjunto de entrenamiento por la red un determinado número de veces dado por t_{fin} y observado que la distorsión ya no disminuye apreciablemente.

Se han publicado diversos algoritmos cercanos al SOFM [Hecht-Nielsen 91], [Yair 92], [Pitas 96], [Zheng 96], diferenciándose unos de otros por la función de adaptación $\alpha(t)$ o la de vecindad $N_j(t)$ y, en último término, por la función de distorsión que tratan de minimizar.

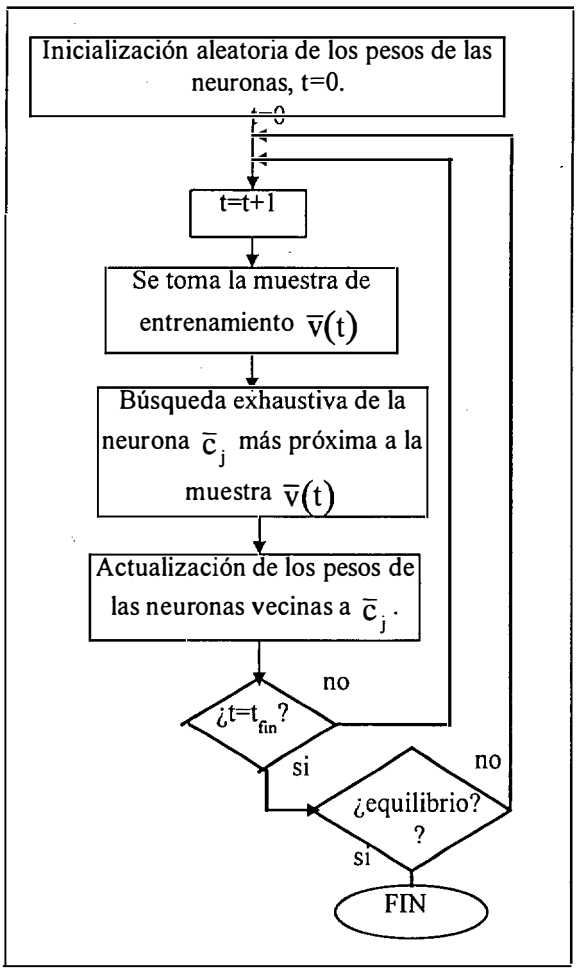

Fig. 3.4.a Mapas Autoorganizativos (diagrama de flujo)

Paso 1. Inicialización.

Paso 2. Avanzar la iteración: t=t+1

Paso 3. Se toma la muestra de entrenamiento $\bar{v}(t)$

Paso 4. Búsqueda exhaustiva de la neurona (vector código) $\bar{c}_j$ más próximo a $\bar{v}(t)$.

Paso 5. Actualización de los pesos de las neuronas vecinas a $\bar{c}_j$ de acuerdo con (3.2.2.3.1).

Paso 6. SI t ha alcanzado la f_{in}
6.1- SI hay equilibrio
PARAR
6.2- SI NO
Volver al Paso 2.

Paso 7. SI t NO ha alcanzado la f_{in}
Volver al Paso 2.

Fig. 3.4.b Mapas Autoorganizativos (pseudocódigo)

b- Aprendizaje Competitivo Simple [Hertz 91]

Mal llamado Cuantificación Vectorial de Aprendizaje (LVQ) por algunos autores(*) es parecido al SOFM, sólo que eliminando la función de vecindades.

En cada iteración sólo es actualizada la neurona $\bar{c}_j$ más próxima a la muestra de entrenamiento $\bar{v}(t)$. La ley de actualización es ahora:

(*) Véase la severa crítica a [Pal 93] que aparece en [González 95].

$$\bar{c}_j(t+1) = \bar{c}_j(t) + \alpha(t)\left(\bar{v}(t) - \bar{c}_j(t)\right), \qquad \textbf{(3.2.2.3.2)}$$

Debido a su carácter local, este algoritmo fácilmente puede quedar atrapado en mínimos locales.

c- Cuantificación Vectorial de Aprendizaje Generalizado [Pal 93]

Es similar al anterior, pero con carácter global: no sólo se optimiza el centroide vencedor, sino todos los demás. Pal propone una ley de actualización que incluye vecindades, pero medidas en el dominio de la señal, no en el mapa de representación:

$$\bar{c}_i(t+1) = \bar{c}_i(t) + \alpha(t)\left(\bar{v}(t) - \bar{c}_i(t)\right), \Psi\left(\bar{v}, \bar{c}_i(t), C(t)\right) \qquad \textbf{(3.2.2.3.3)}$$

donde $\Psi\left(\bar{v}, \bar{c}_i(t), C(t)\right)$ es la función de vecindad que depende no sólo de la muestra actual de entrenamiento y del centroide (vector código) que se va a actualizar, sino del resto de los centroides de la librería.

En esta misma línea se encuentran la Red "Neural-Gas" [Martínez 93] y el algoritmo SCONN [Choi 94].

3.2.2.4- Métodos "Fuzzy"

La Cuantificación Vectorial Fuzzy [Karayiannis 95] se basa en hacer que las decisiones de inclusión de un vector de entrenamiento a una subpartición sean "blandas", esto es, un vector pertenece a una subpartición, pero de una forma sólo parcial. Formalmente, estos vínculos vienen determinados por la función $\zeta(\bar{v}, j)$ de pertenencia del vector $\bar{v}$ a la subpartición S_j, que toma valores en el intervalo [0,1] (mayores cuanto más próximo esté $\bar{v}$ de $\bar{c}_j$).

La función global que se trata de minimizar es ahora:

$$D = \sum_{i=0}^{L-1} \sum_{j=0}^{N-1} \zeta(\bar{v}_i, j) \left\| \bar{v}_i - \bar{c}_j \right\|_2 \qquad \textbf{(3.2.2.4.1)}$$

Recordando que, para el caso de que las características de la fuente estuvieran contenidas en el conjunto de entrenamiento $V = \{\bar{v}_0, \bar{v}_1, .., \bar{v}_{N-1}\}$, la función de distorsión podía ponerse como la suma de las distorsiones parciales D_i asociadas a cada región (2.1.3.4.4):

$$D = \sum_{i=0}^{L-1} D_i = \sum_{i=0}^{L-1} \sum_{j=0}^{N_i-1} \left\| \bar{v}_{ij} - \bar{c}_i \right\|_2 \qquad \textbf{(3.2.2.4.2)}$$

puede observarse que ambas expresiones coinciden plenamente, salvo por la función "fuzzy" $\zeta(\bar{v}, j)$ de pertenencia a las regiones. Más aún, en el caso extremo en que esta función diera valor 1 cuando $\bar{v}$ estuviera contenido en la región j, y 0 en caso contrario, la dos distorsiones serían exactamente iguales. En este caso, se hablará de partición del espacio y reglas de decisión "duras".

Las decisiones del algoritmo van pasando de blandas a duras según la estrategia que se describe a continuación. En el instante de tiempo t, el vector $\bar{v}_i$ tiene asignado una hiperesfera en la cual se encuentran varios centroides (vectores código). Al conjunto de esos centroides se le llama $I_i(t)$. La gradual transición hacia una partición dura se consigue disminuyendo el radio de esta hiperesfera. En cada iteración este radio se recalcula, como la media de las distancias entre $\bar{v}_i$ y los centroides incluidos en $I_i(t)$:

$$d(\bar{v},t) = \frac{1}{N(I_i(t))} \sum_{\bar{c}_j \in I_i(t)} \left\| \bar{v}_i - \bar{c}_j \right\|_2 \qquad \textbf{(3.2.2.4.3)}$$

donde $N(I_i(t))$ es el número total de elementos contenidos en el conjunto $I_i(t)$

El conjunto $I_i(t)$ se actualiza según:

$$I_i(t) = \left\{ \bar{c}_j \in I_i(t) \;\middle|\; \left\| \bar{v}_i - \bar{c}_j \right\|_2 \leq d(\bar{v},t) \right\} \qquad \textbf{(3.2.2.4.4)}$$

Este proceso comienza con todo el espacio

$$I_i(0) = C \qquad \textbf{(3.2.2.4.5)}$$

y acaba cuando sólo hay un centroide en la hiperesfera.

En cada iteración los vectores código se calculan según:

$$\bar{c}_i = \frac{\sum_{j=0}^{N_i - 1} \zeta(\bar{v}_i, j)\, \bar{v}_j}{\sum_{j=0}^{N_i - 1} \zeta(\bar{v}_i, j)} \qquad \textbf{(3.2.2.4.6)}$$

que viene a ser la versión "fuzzy" de la Regla de los Centroides.

En la anterior referencia pueden encontrase distintas variantes de esta ley, así como de las funciones de vecindad $\zeta(\bar{v}, j)$.

En la misma línea se hallan los algoritmos expuestos por el mismo autor en [Karayiannis 96] o en [Karayiannis 97].

3.3- Optimización en CV con errores de canal

Según se explicaba en la sección 2.1.4, cuando existen errores en el canal, la distorsión depende del proceso de cuantificación y del error cometido en la transmisión de los símbolos. Ello conduce a que esta distorsión sea función del libro de códigos (regla de decodificación), de la partición del espacio (regla de codificación) y de la función de asignación.

Veamos, a continuación, tres planteamientos diferentes para optimizar el sistema.

3.3.1- Reordenamiento de índices

El planteamiento aquí es partir de una librería de códigos diseñada para el caso no ruidoso y optimizar la asignación de índices, de tal forma que los errores de símbolos produzcan distorsiones mínimas en el sistema.

En el Apartado 2.1.4.2 se observó que si los vectores código se elegían de acuerdo con la Regla de los Centroides no generalizada, la distorsión de un cuantificador vectorial sujeto a errores de canal podía descomponerse en dos términos independientes correspondientes a la distorsión de fuente y a la distorsión de canal:

$$D = D_F + D_C \qquad \textbf{(3.3.1.1)}$$

También se veía que incluso sin estar presente esta regla, la distorsión total del sistema está acotada por la suma de ambas distorsiones [Zeger 87]:

$$D \leq D_F + D_C \qquad \textbf{(3.3.1.2)}$$

Como la D_F no depende de las asignaciones de índices, tiene sentido tratar de minimizar sólo el término D_C Como expresa Zeger en el artículo correspondiente a la referencia anterior: "este planteamiento no garantiza necesariamente la minimización de la distorsión total, pero es una solución heurística que puede contribuir a su reducción".

Según esto y volviendo a la nomenclatura definida en el apartado 2.1.4.1, la variable de búsqueda será la función de asignación Π, expresada por la lista

$$B_\Pi = \{\Pi(0), \Pi(1), \quad , \Pi(L-1)\} \qquad \textbf{(3.3.1.3)}$$

que indica la relación biunívoca entre vectores código y los símbolos transmitidos por el canal. Según se indicaba también en el anterior apartado, esta lista es una de las L! permutaciones posibles de la lista correspondiente a la "asignación natural" dada por:

$$B_{\Pi_N} = \{0, 1, \quad , L-1\} \qquad \textbf{(3.3.1.4)}$$

La distorsión de canal dada en (2.1.4.2.11) depende de las probabilidades de transmisión P_{ji}, y éstas, a través de (2.1.4.3.4), de la función de asignación presente. Como, por otra parte, según la estrategia aquí considerada, el libro de códigos permanece fijo, esta distorsión puede considerarse una función únicamente de la variable de asignación Π:

$$D_C = D_C(\Pi) \qquad \textbf{(3.3.1.5)}$$

Existen diversas aproximaciones dentro de esta línea que trata de minimizar la distorsión D optimizando solamente la asignación Π, algunas de las cuales se comentan a continuación.

3.3.1.1- Métodos heurísticos

a- Codificación Pseudo-Gray [Zeger 87], [Zeger 90]

Es éste un algoritmo iterativo que va permutando la posición de dos vectores código en cada iteración, intentando disminuir monótonamente la función $D_C(\Pi)$. Para elegir qué pareja va a ser permutada, se asigna un coste a cada vector código:

$$D_i = P_i \sum_{j=0}^{L-1} P_{ji} \left\| \bar{c}_{ij} - \bar{c}_j \right\|_2 \qquad \textbf{(3.3.1.1.1)}$$

que mide la contribución total a esta distorsión, debida a posibles errores en el canal cuando se ha seleccionado el vector $\bar{c}_i$.

Todo el libro de código es ordenado de acuerdo con este coste. El vector código con mayor D_i (llamémosle $\bar{y}_0$) es elegido como primer candidato para la permutación. Se hace entonces una permutación tentativa entre éste y todos los demás vectores, observándose el potencial decremento de $D_C(\Pi)$ con cada uno de ellos. Aquel vector correspondiente al mayor decremento de $D_C(\Pi)$ es con quien $\bar{y}_0$ se permuta. Si no disminuye con ninguno de ellos, le llega al turno al segundo vector con mayor D_i, $\bar{y}_1$, al que se le intenta permutar de la manera anterior. Si tampoco se puede conseguir disminuir la distorsión total, se probará con el siguiente vector, el $\bar{y}_2$, y así sucesivamente. Se llega al final a un estado localmente mínimo en el que no existe ninguna posible permutación de un par de vectores que conduzca a un estado de menor distorsión media.

Este algoritmo, que permite asignar índices próximos (con distancias hamming pequeñas) a vectores código cercanos en el espacio euclídeo, fue bautizado por sus autores como Codificación Pseudo-Gray, por analogía con los códigos Gray usados en cuantificación escalar, con los que las muestras contiguas son representadas con cadenas binarias que sólo se diferencian en un bit. El organigrama del algoritmo se muestra en la Figura 3.5.

En [De Marca 87] se expone un algoritmo parecido a éste, pero en él se introducen perturbaciones locales a las asignaciones tentativas, tratando de evitar mínimos locales.

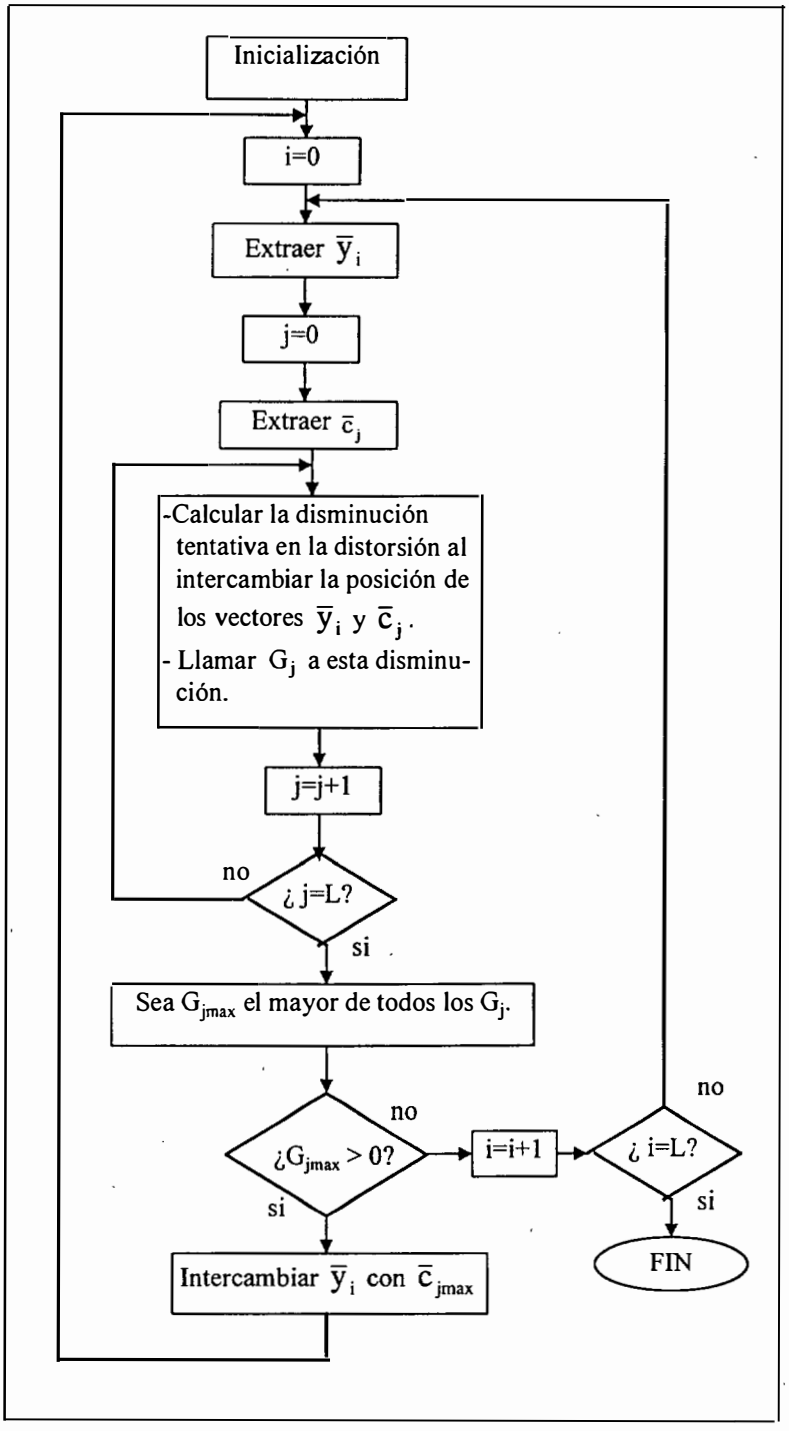

Fig. 3.5 Algoritmo de Codificación Pseudo-Gray (diagrama de flujo)

b- Solución de Wu y Barba

En [Wu 93] estos autores proponen otra solución basada en crear un árbol de índices binarios. Los nodos de cada nivel del árbol corresponden a índices diferentes. Cada nodo está unido a varios nodos del nivel anterior, exactamente, a aquéllos cuyos índices distan una unidad de éste (según la distancia hamming). La figura muestra el

caso de 16 niveles. Los nodos están marcados por los puntos en negro y van acompañados de enteros del 0 al 15 representando códigos binarios de 4 bits (para mayor claridad sólo se muestran los binarios correspondientes al 12 y al 14).

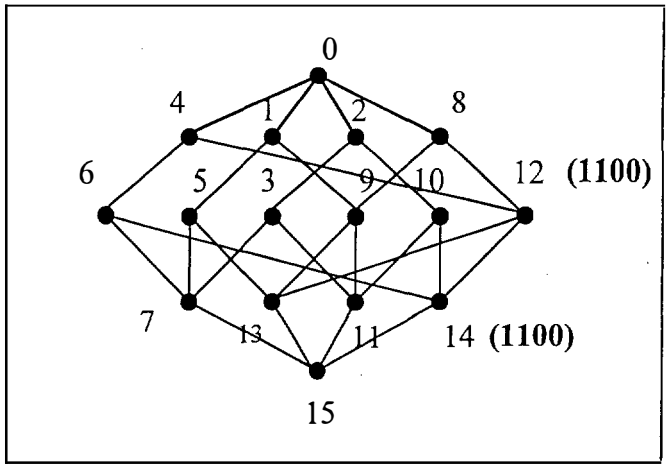

Fig.3.6 Arbol topográfico del algoritmo de Wu y Barba.

Al vector código más probable se le asigna el primer nodo. A los vectores más próximos a éste se les asignan los nodos del segundo nivel. A cada nodo de un nivel inferior se le asigna un vector que esté próximo en el espacio euclídeo a aquéllos vectores asociados a los nodos del nivel anterior, unidos directamente con este nodo.

Además el cálculo de distancias anterior se pondera por la probabilidad de aparición "a priori" de un vector código (P_i), de tal manera que un vector código con alta probabilidad de aparición tiende a ser asignado a altos niveles, para que así tenga más libertad para ser elegido vecino de los vectores próximos a él.

En particular, para elegir el vector código asociado al nodo k del nivel v, se tendrán en cuenta todos los vectores $\bar{c}_i$ pertenecientes al conjunto $S_{v-1,k}$, de todos los vectores del nivel v-1 que son vecinos de k. Para cada vector $\bar{c}_j$, potencial ocupante del nodo k se establecerá una medida de distorsión parcial (medida conjunta de la distancia entre éste y cada uno de los vectores de $S_{v-1,k}$, junto con sus probabilidades de aparición):

$$\eta(k,j) = \sum_{\bar{c}_i \in Sv-1,k} \frac{\left\| \bar{c}_i - \bar{c}_j \right\|_2}{P_i + P_j} \qquad \textbf{(3.3.1.1.2)}$$

entre los vectores $\bar{c}_j$ que todavía no han ocupado ningún nodo, se elegirá aquél que cuente con menor distorsión parcial $\eta(k,j)$.

Paso 1. Formar el árbol topográfico.

Paso 2. Buscar el vector código más probable y asignarle el nodo 0.

Paso 3. Computar la distancia euclídea entre éste y el resto de los vectores código. Elegir Los b más próximos a él para que ocupen los nodos del nivel 1.

Paso 4. Sea el nivel v=2

Paso 4. Cada nodo del nivel v tiene v-1 vecinos en el nivel v-1. Para cada nodo k del nivel v y todos los vectores $\bar{c}_j$ todavía no asignados a ningún nodo, se calcula la distorsión parcial $\eta(k,j)$ dada en (3.3.1.8).

Paso 5. Se asocia al nodo k el vector $\bar{c}_j$ que haga menor esta distorsión.

Paso 6. SI v<b
6.1- Pasar al nivel siguiente: v=v+1
6.2- Volver al Paso 4.

Paso 7. SI NO
PARAR

Fig. 3.7 Algoritmo de Wu y Barba (pseudocódigo)

3.3.1.2- Métodos termoestadísticos

El Temple Simulado, descrito en le apartado 3.2.2.1 para hallar libros de código también puede usarse para encontrar funciones de asignación que reporten bajas distorsiones [El Gamal 87], [Farvardin 90]. Las variables en juego serán las funciones de asignación Π, expresadas con las listas B_Π y la función optimizada, será la codificación de canal C_F, según se explicaba en el Apartado 3.3.1.

La perturbación aleatoria de las listas B_Π se realizará intercambiando una pareja de sus elementos, de forma semejante al operador mutación aplicado a la ordenación de listas, que aparecía en 2.2.6.3.a. También podría aplicarse cualquier otra mutación de las expuestas en ese apartado.

3.3.1.3- Métodos genéticos

Pan et. al. describen un AG para optimizar la asignación de índices [Pan 96]. Los elementos distintivos de este algoritmo son:

- Individuos de la población: son las funciones Π, también expresadas como listas B_Π;
- Población: está separada en varios grupos que evolucionan de forma independiente, salvo por el operador "comunicación", del que se hablará más abajo;

- Función de coste: distorsión del canal D_c.
- Selección: es por torneo y se realiza en cada grupo por separado;
- Cruce: ordenado uniforme;
- Mutación: por intercambio recíproco;
- Comunicación: cada R generaciones se envían los mejores individuos de un grupo a otros grupos reemplazando a individuos elegidos al azar.

3.3.2- Optimización conjunta de vectores e índices

El algoritmo GLA descrito en el Apartado 2.1.4.5 es un método elegante y de reducida complejidad conceptual y de implementación. Al utilizar consecutiva y alternadamente las dos Reglas de Optimalidad el proceso de búsqueda avanza por pasos que reducen la distorsión de forma no sólo monótona, sino óptima. Sin embargo, la librería resultante es sólo localmente óptima; son necesarios métodos más globales de búsqueda. Gran parte de estos otros métodos llevan a cabo una exploración global e incorporan el GLA para concluir de forma "fina" la búsqueda.

En este apartado se recogen varios de estos métodos, que tratan de minimizar la distorsión sin realizar un tratamiento diferenciado para la distorsión de canal y la de fuente. Su estrategia es la de reajustar las posiciones del conjunto inicial de vectores código, para hacer disminuir la distorsión media, teniendo presente la representación binaria inicial de cada uno de estos vectores y el juego de probabilidades de error P_{ij}. Aunque explícitamente no lo hacen, implícitamente sí consideran y tratan de minimizar los errores de canal. De hecho, el algoritmo GLA, aunque de una forma local, atiende a estos planteamientos.

3.3.2.1- Métodos termoestadísticos

El Temple Determinístico (TD) de Miller [Miller 94] es una combinación del Temple Simulado y de los métodos "fuzzy". La diferencia con el TS radica en que éste introduce variaciones aleatorias sobre los puntos donde la función de coste es evaluada, mientras que el TD introduce un ruido directamente sobre la función de coste (distorsión media).

Expondremos el TD sólo para el diseño basado en datos empíricos, aunque su extensión al caso continuo, en que los estadísticos están dados por la f.d.p. de la fuente, es clara.

La Regla del VMP establece una partición "dura" del espacio de vectores de entrenamiento, por la que cada vector sólo se relaciona con la región a la que pertenece. Esta relación puede encarnarse en la función de asociación fuerte entre el vector de entrenamiento $\bar{v}$ y la subpartición V_j, dada por:

$$A_f(\bar{v}, j) \quad \begin{cases} 1, & \text{si } \bar{v} \in V_j \\ 0, & \text{resto} \end{cases} \qquad \textbf{(3.3.2.1.1)}$$

Sin embargo, puede definirse una función de asociación débil entre el vector $\bar{v}$ y la región V_j, según la cual, la relación entre ambos vaya disminuyendo cuanto más alejada se encuentre $\bar{v}$ de V_j. Esta función de asociación débil hará las veces de regla de codificación, difuminando el concepto de partición del espacio de vectores.

La propuesta de este método es el progresivo paso de asociaciones blandas a duras en el transcurso de las iteraciones. Para ello se cuenta con un parámetro β que va aumentando con las iteraciones. Las funciones de asociación se construyen para que, en los casos límite en que β sea cero o tienda a infinito, se cumpla que:

$$\lim_{\beta \to 0} \left\{ A_d(\bar{v}, j, \beta) \right\} = \frac{1}{L} \qquad \forall\, \bar{v} \in V \tag{3.3.2.1.2}$$

$$\lim_{\beta \to \infty} \left\{ A_d(\bar{v}, j, \beta) \right\} = \begin{cases} 1, & \text{si } \bar{v} \in V_j \\ 0, & \text{resto} \end{cases} \tag{3.3.2.1.3}$$

con lo cual, cuando β=0 cualquier vector del conjunto de entrenamiento estará igualmente relacionado con cualquier región (puede pensarse que de hecho el espacio no está dividido en regiones). En cambio, cuando $\beta \to \infty$, la asociación débil se convierte en fuerte y el espacio queda concretamente dividido por la partición.

Haciendo que β crezca a lo largo de las iteraciones del algoritmo, se conseguirá un progresivo endurecimiento de la partición.

Una función de asociación débil basada en el Método de Máxima Entropía [Jaynes 89] puede ser:

$$A_d(\bar{v}, j, \beta) = \frac{e^{-\beta \sum_{k=0}^{L-1} P_{jk} \left\| \bar{v} - \bar{c}_k \right\|_2}}{\sum_{r=0}^{L-1} e^{-\beta \sum_{k=0}^{L-1} P_{rk} \left\| \bar{v} - \bar{c}_k \right\|_2}} \tag{3.3.2.1.4}$$

que cumple las características descritas para estas funciones, incluídas las condiciones límite anteriores.

Recordemos ahora la Regla Generalizada de los Centroides y veamos de qué manera puede incorporar los nuevos conceptos de asociación débil y partición blanda. La RGC está dada por la expresión (2.1.4.6.7):

$$\bar{c}_i \quad \frac{\sum_{j=0}^{L-1} P_{ji} \cdot P_j \cdot \hat{c}_j}{\sum_{j=0}^{L-1} P_{ji} \cdot P_j} \qquad i=0, 1, \ldots L\text{-}1 \tag{3.3.2.1.5}$$

donde P_j es la probabilidad de que el vector $\bar{v}$ se encuentre en la región $V_j = \left\{ \bar{v}_{j1}, \bar{v}_{j2}, \ldots, \bar{v}_{j N_j - 1} \right\}$ que, según (2.1.4.6.8) es:

$$P_j = \frac{N_j}{N} \tag{3.3.2.1.6}$$

y el centroide $\hat{c}_j$ es (2.1.3.4.7):

$$\hat{c}_j = \frac{1}{N_j} \sum_{p=0}^{N_j-1} \bar{v}_{jp} \qquad \textbf{(3.3.2.1.7)}$$

Uniendo las tres últimas expresiones, la RGC queda:

$$\bar{c}_i = \frac{\sum_{j=0}^{L-1} P_{ji} \cdot \sum_{p=0}^{N_j-1} \bar{v}_{jp}}{\sum_{j=0}^{L-1} P_{ji} \cdot N_j} \qquad \textbf{(3.3.2.1.8)}$$

que también puede escribirse, atendiendo a la definición de asociación dura de (3.3.2.1.1) como:

$$\bar{c}_i = \frac{\sum_{j=0}^{L-1} P_{ji} \cdot \sum_{p=0}^{N-1} \bar{v}_p A_f\left(\bar{v}_p, j\right)}{\sum_{j=0}^{L-1} P_{ji} \cdot \sum_{p=0}^{N-1} A_f\left(\bar{v}_p, j\right)} \qquad \textbf{(3.3.2.1.9)}$$

Si se sustituye la asociación fuerte por la asociación blanda, quedará:

$$\bar{c}_i = \frac{\sum_{j=0}^{L-1} P_{ji} \cdot \sum_{p=0}^{N-1} \bar{v}_p A_d\left(\bar{v}_p, j, \beta\right)}{\sum_{j=0}^{L-1} P_{ji} \cdot \sum_{p=0}^{N-1} A_d\left(\bar{v}_p, j, \beta\right)} \qquad \textbf{(3.3.2.1.10)}$$

Esta asociación, junto con la expresión (3.3.2.1.4), aplicadas sucesivas veces de forma alternada, con un valor determinado de β llevarán al sistema a un mínimo de energía local. Haciendo que β tome valores cada vez más altos, el algoritmo irá conduciendo al sistema hacia una partición dura del espacio de vectores.

En [Miller 94] no aparece la ley con la que β va aumentando. Tampoco se indica en [Rose 92] (versión no ruidosa del TD). En [Miller 96], en el contexto del diseño de clasificadores estadísticos, se presenta una sencilla regla para la evolución temporal de este parámetro:

$$\beta(t) = \alpha\, \beta(t-1) \qquad \textbf{(3.3.2.1.11)}$$

En la Figura 3.8 se muestra el diagrama funcional y el pseudocódigo del algoritmo de TD explicado.

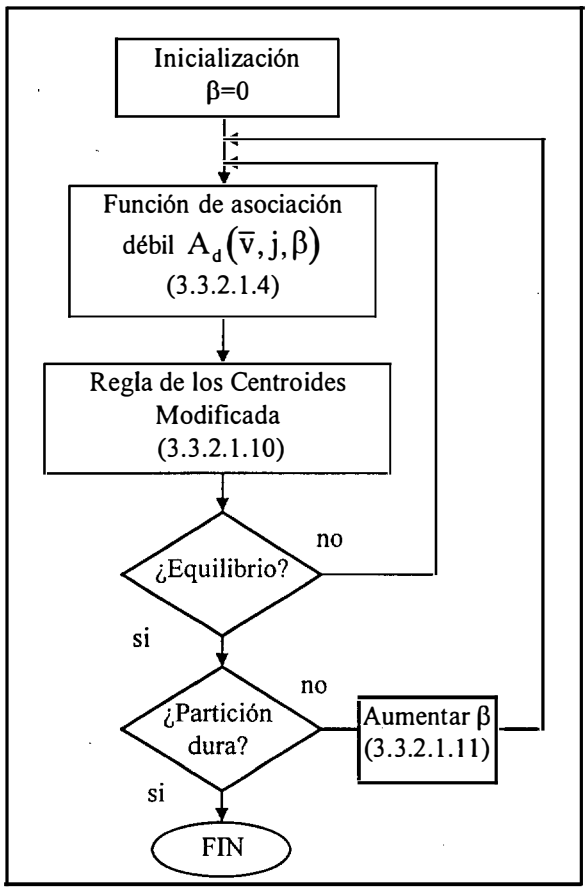

Fig. 3.8.a Temple Determinístico (diagrama de flujo)

Paso 1. Inicialización.

Paso 2. Calcular las funciones de asociación débil dadas en (3.3.2.1.4)

Paso 3. Computar la Regla de los Centroides Modificada según (3.3.2.1.10)

Paso 4. SI NO hay equilibrio
Volver al Paso 2.

Paso 5. SI hay equilibrio
6.1- SI se ha llegado a una partición dura
PARAR
6.2- SI NO
Aumentar el valor de β según (3.3.2.1.11).
VOLVER al Paso 2.

Fig. 3.8.b Temple Determinístico (pseudocódigo)

3.3.2.2- Métodos basados en Redes Neuronales Autoorganizativas

Resultan de la natural generalización al caso ruidoso de los algoritmos SOFM vistos en 3.2.2.3. Ahora las relaciones topológicas entre las neuronas (vectores código) en el dominio representado pueden ligarse a las relaciones de cercanía o lejanía de los símbolos (binarios o no) con los que esos vectores serán enviados.

a- En el TVQ [Black 93] la ley de actualización del vector de código $\bar{c}_j$ cuando se le presenta el vector código $\bar{v}(t)$ es:

$$\bar{c}_j(t) = \bar{c}_j(t-1) + \varepsilon \cdot P_{ji} \cdot \left(\bar{v}_j(t) - \bar{c}_j(t-1)\right) \qquad \textbf{(3.3.2.2.1)}$$

donde i es el índice del vector código más próximo a $\bar{v}(t)$ y ε, un parámetro del método similar al paso de adaptación en los algoritmos adaptativos.

b- Una propuesta muy similar se detalla en [Knagenhjelem 92]. Aquí ε es sustituido por una función que varía con las iteraciones, de forma que las actualizaciones cada vez tienen influencia sobre vecindades menores (más cercanas al vecino más próximo del vector de entrenamiento $\bar{v}(t)$ inspeccionado).

c- Otra versión muy similar a las anteriores es la dada en [Poggy 95], en la que el factor P_{ji} es sustituido por una función de la distancia hamming entre los índices i y j.

d- En [Leung 97] los anteriores conceptos se extienden al caso de modulaciones genéricas, de las que la transmisión de símbolos binarios es sólo un caso particular.

3.3.3- Optimización alternada de vectores e índices

Cuperman utiliza una metodología de diseño basada en considerar separadamente los tres elementos del CV: el codificador (partición Ω), el decodificador (librería de códigos $\{C\}$) y la función de asignación Π. Cada uno de ellos se optimiza, dejando fijos los otros dos elementos, de manera alternada e iterativa, como aparece en el diagrama de bloques de la Figura 3.9.

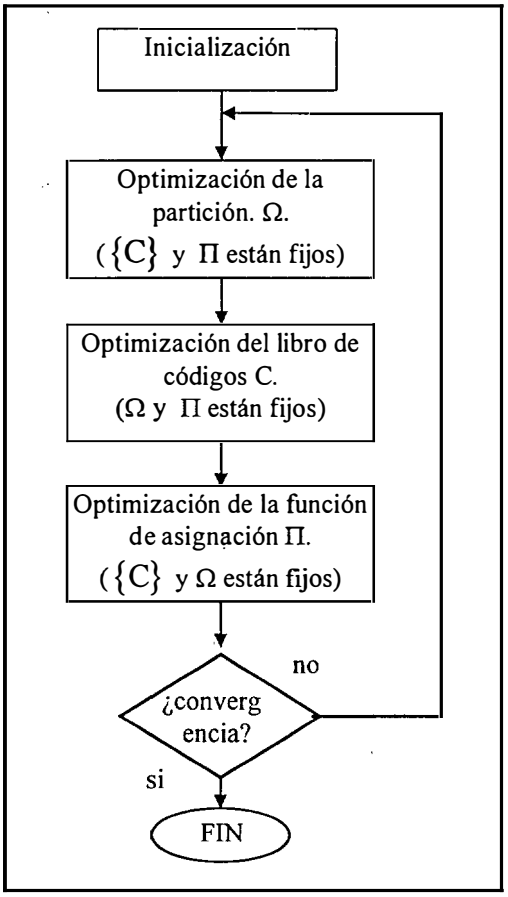

Fig. 3.9.a Algoritmo de optimización VQ de Cuperman (diagrama de bloques)

Paso 1. Inicialización.

Paso 2. Optimización de la partición Ω.

Paso 3. Optimización del libro de códigos $\{C\}$.

Paso 4. Optimización de la función de asignación Π.

Paso 5. SI hay equilibrio
PARAR

Paso 6- SI NO
VOLVER al Paso 2.

Fig. 3.9.b Algoritmo de optimización VQ de Cuperman (pseudocódigo)

La optimización de la partición y de la librería de códigos no coinciden exactamente con las reglas RGVMP y RGC, debido a que el planteamiento del problema que hace Cuperman es ligeramente diferente presentado en el apartado 2.1.4.1. El decodificador de Cuperman no entregará a la salida uno de los vectores código de la librería, sino una combinación lineal de ellos, que tenga en cuenta la f.d.p. del ruido en el canal. El resto del planteamiento y la forma de derivar sus reglas de optimalidad son semejantes a lo aquí explicado.

Capítulo 4

PROPUESTAS

4.1- Introducción

En el apartado 2.1.4.1, la CV se desglosaba en distintos bloques funcionales dibujados en la Figura 2.22, que repetimos aquí, para mayor claridad.

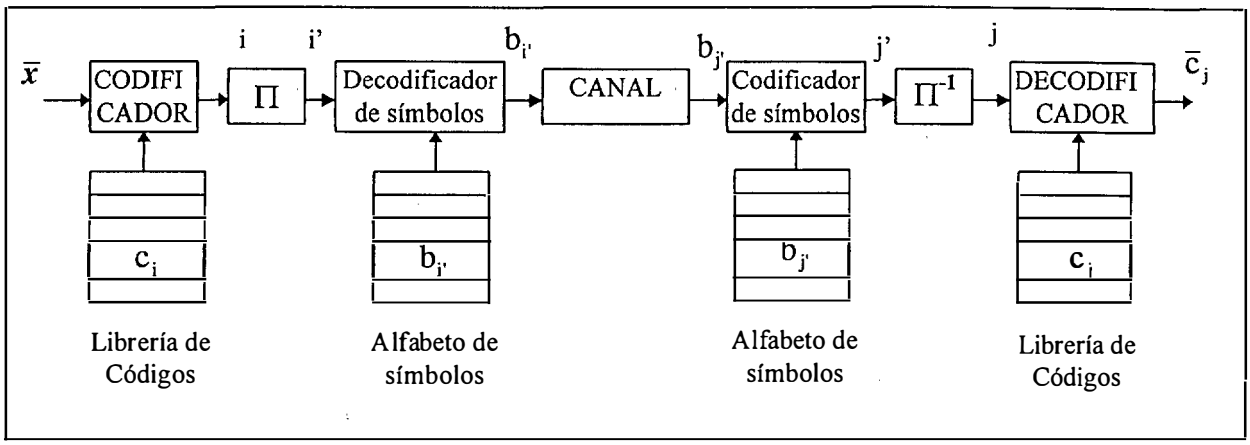

Fig. 4.1 Sistema de Cuantificación Vectorial

El propósito general en el diseño es el de ajustar los elementos del sistema para que la distorsión media entre la entrada y la salida

$$D = E\left\{\left\|\bar{x} - \bar{c}_j\right\|_2\right\} \qquad \textbf{(4.1.1)}$$

sea mínima (ver apartado 2.1.4.2).

En nuestro caso, la fuente estará caracterizada estadísticamente por el conjunto de vectores de entrenamiento

$$V = \left\{\bar{v}_0, \bar{v}_1, \ , \bar{v}_{N-1}\right\} \qquad \textbf{(4.1.2)}$$

por lo que la distorsión del proceso será la expresada en (2.1.4.6.4):

$$D = \sum_{i=0}^{L-1} D_i = \frac{1}{N} \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} \sum_{k=0}^{N_i-1} P_{ji} \left\| \bar{v}_{ik} - \bar{c}_j \right\|_2 \qquad \textbf{(4.1.3)}$$

donde $\bar{v}_{ik}$ es un elemento genérico de la subpartición V_i expresada como:

$$V_i = \{\bar{v} \in V / Cod(\bar{v}) = i\} = \{\bar{v}_{ik}, k = 0, 1, .., N_i - 1\} \qquad \textbf{(4.1.4)}$$

En principio, los elementos variables y, por tanto, optimizables del sistema son:
- la regla de codificación dada por la partición $\{V\}$;
- la librería de códigos $\{C\}$;
- la función de asignación Π;
- el alfabeto de símbolos B

Nótese que el codificador y el decodificador de símbolos son estructuras fijas, no sujetas a optimización (ver Figuras 2.20 y 2.21).

Por otra parte, convendremos que la codificación es binarias y el alfabeto B sigue la ordenación natural dada en (2.1.4.3.2).

Por último, supondremos que todos nuestros CV atienden a la RGVMP, por lo que la partición $\{V\}$ deja de ser una variable independiente del sistema al estar completamente determinada por el resto de variables del sistema (ver apartado 2.1.4.4.1).

Según todo lo anterior, sólo se contará con $\{C\}$ y Π para optimizar el sistema, lo cual equivale a decir que la distorsión D dependerá únicamente de estas dos variables:

$$D = D(\{C\}, \Pi) \qquad \textbf{(4.1.5)}$$

En esta Tesis se describen tres métodos de optimización de CV. Los dos primeros tratan de llevar a cabo una optimización explícita del libro de códigos. La asignación Π se mantiene fija, aunque la búsqueda en el dominio de las asignaciones se realiza de forma implícita. Según la clasificación establecida en el apartado 3.3, estos métodos podrían catalogarse como de optimización conjunta de vectores e índices. De los dos, uno está basado en Algoritmos Genéticos y el otro es un método heurístico iterativo cercano al algoritmo de Lloyd.

El tercero de los métodos, sin embargo, mantiene una búsqueda explícita de $\{C\}$ y de Π, dentro de la clase que se ha denominado de optimización alternada de vectores e índices. Este algoritmo también está basado en los AG y se híbrida con el GLA para llevar a cabo la búsqueda.

4.2- Optimización conjunta de vectores e índices basada en Algoritmos Genéticos

Se propone un nuevo algoritmo para el diseño de cuantificadores vectoriales para canales ruidosos en el que un Algoritmo Genético sirve de base en la búsqueda de librería de códigos óptima, al cual se incorpora el algoritmo GLA como optimizador local. Llamaremos a este método AGCV (Algoritmo Genético para la Cuantificación Vectorial)

4.2.1- Descripción

Este AG está especialmente pensado para diseñar CV. Los T individuos de la población Φ son librerías de código tentativas, cada uno representado con una matriz de dimensión LxJ (L vectores código de dimensión J):

$$\Phi=\{\{C_1\},\{C_2\}, \ ,\{C_T\}\} \qquad \textbf{(4.2.1.1)}$$

donde

$$\{C\}_i = \begin{bmatrix} \bar{c}_0^i \\ \bar{c}_1^i \\ \\ \bar{c}_{L-1}^i \end{bmatrix} = \begin{bmatrix} c_{0,0}^i, & c_{0,1}^i, & .. & c_{0,J-1}^i \\ c_{1,0}^i, . & c_{1,1}^i, & .. & c_{1,J-1}^i \\ . \ . & .. & .. & .. \\ c_{L-1,0}^i, & c_{L-1,1}^i, & .. & c_{L-1,J-1}^i \end{bmatrix} \qquad \textbf{(4.2.1.2)}$$

La función de coste es la distorsión media dada en (2.1.4.6.4). Se supondrá además una asignación binaria natural B_{Π_N} (ver apartado (2.1.4.1)), por lo que la distorsión sólo será función explícita de la librería de códigos:

$$D = D(\{C\}) \qquad \textbf{(4.2.1.3)}$$

El AGCV es similar al Genético presentado en [Choi 96], pero extendido para abarcar el caso del canal ruidoso. El algoritmo presenta diferentes alternativas con respecto a los mecanismos de selección, cruce, estrategias de evolución y tratamiento de clones. Todas ellas se describen a continuación, junto con el operador de mutación usado y el elitismo incorporado.

4.2.1.1- Mecanismos de selección

a- Selección Aleatoria Proporcional a la Función de Prestaciones:

En una generación dada, la función de prestaciones F( ) de cada individuo i (libro $\{C\}_i$) está dada por la diferencia entre la distorsión mayor encontrada en toda la población y la distorsión asociada a este individuo:

$$F(\{C\}_i) = \underbrace{\max}_{k=1, \ ,T} \{D(\{C\}_k)\} - \{D(\{C\}_i)\} \qquad \textbf{(4.2.1.1.1)}$$

De acuerdo con la expresión (2.2.4.6), la probabilidad de que la librería $\{C\}_i$ resulte elegida en cualquier proceso de selección es:

$$p(i) = \frac{F(\{C\}_i)}{\sum_{k=1}^{T} F(\{C\}_k)} \qquad \textbf{(4.2.1.1.2)}$$

b- Selección por Rango:

Según lo explicado en (2.2.4.b), las distorsiones relacionadas con los T individuos de la población se ordenan de mayor a menor de la siguiente forma: al individuo con menor distorsión se le asignará orden 1, y al de mayor distorsión, se le asignará orden T. Después estos órdenes son transformados linealmente de tal forma que el orden 1 se trasforma en P_{min} y el orden T, en P_{max}. Finalmente se procede al la selección aleatoria proporcional a estos valores. En nuestro caso, los parámetros P_{min} y P_{max} toman los valores 1 y 5, respectivamente.

c- Selección por Torneo:

Según lo explicado en (2.2.4.c), se seleccionan aleatoriamente dos individuos de la población y aquél con menor distorsión es seleccionado.

4.2.1.2- Mecanismos de cruce

Se incluyen tres operadores diseñados de forma específica para el cruce entre librerías de código.

a- Cruce Aleatorio:

Es similar al cruce aritmético presentado en el apartado 2.2.5.2.b. Tras haber sido seleccionados, el cruce entre dos individuos se realiza vector a vector. En concreto, si $\bar{c}_p^i$ (vector p del libro de códigos $\{C\}_i$) va a ser cruzado con $\bar{c}_q^j$ (vector q del libro de códigos $\{C\}_j$), el nuevo vector será la media aleatoria entre $\bar{c}_p^i$ y $\bar{c}_q^j$ definida como:

$$ra\left(\bar{c}_p^i, \bar{c}_q^j\right) = \bar{c}_p^i - \frac{\overline{dif}}{2} + 2\, r \cdot \overline{dif} \qquad \textbf{(4.2.1.2.3)}$$

donde $\overline{dif}$ es el vector diferencia entre los vectores código $\bar{c}_p^i$ y $\bar{c}_q^j$:

$$\overline{dif} = \bar{c}_q^j - \bar{c}_p^i \qquad \textbf{(4.2.1.2.4)}$$

y r es un vector de dimensión M, en el que cada una de sus componentes se toma de forma completamente aleatoria en el intervalo [0,1]. De esta forma, el nuevo vector se encontrará de forma equiprobable en el intervalo M-dimensional $\left[c_p^i - \frac{dif}{2},\ \bar{c}_q^j + \frac{dif}{2} \right]$ (ver la Figura 4.2, que muestra un ejemplo en dos dimensiones).

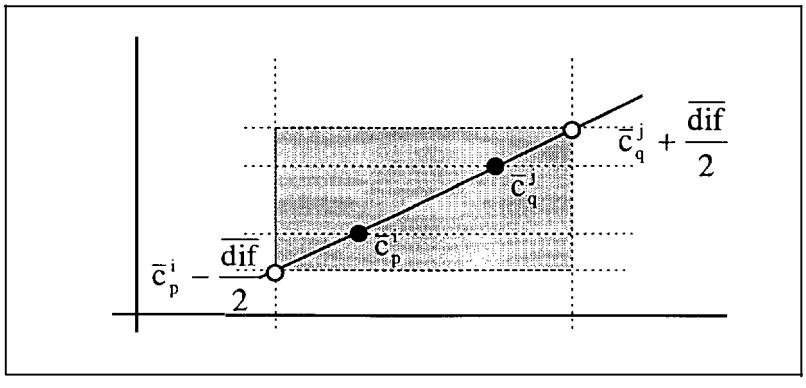

Fig. 4.2 Cruce aleatorio. El vector resultante del cruce entre los vectores $\bar{c}_p^i$ y $\bar{c}_q^j$ se encontrará en la región sombreada.

Según esto, cuando se cruzan $\{C\}_i$ y $\{C\}_j$ se obtiene un nuevo $\{C\}_k$ en el que los vectores código están dados por:

$$\bar{c}_p^k = ra\left(\bar{c}_p^i, \bar{c}_p^j\right) \quad , \qquad p=0,1,\ L\text{-}1 \qquad \textbf{(4.2.1.2.5)}$$

b- Cruce Constreñido:

En el cruce anterior, el simple procedimiento de cruzar pares de vectores código de dos librerías de la población puede no tener demasiado sentido si estos vectores están muy separados entre sí. Para entender esta circunstancia en la Figura 4.3 se muestra un sencillo ejemplo el que se cruzan dos librerías de 5 vectores de dimensión 2.

Se dibujan los vectores códigos de $\{C\}_i$ (Fig 4.3. izquierda) y de $\{C\}_j$ (Fig 4.3. derecha), junto con sus respectivas particiones. Como los vectores en una librería no están ordenados atendiendo a su localización espacial, no es sensato presuponer que el vector $\bar{c}_p^i$ vaya a tener mayor proximidad al vector $\bar{c}_p^j$ que a ningún otro $\bar{c}_q^j$ de $\{C\}_j$. El cruce aleatorio entre las parejas $\bar{c}_p^i$ y $\bar{c}_p^j$ (p=0,1, L-1) se tornará un operador demasiado caótico, en el que la estructura de los individuos i y j (ya sean percibidos como las librerías $\{C\}_i$ y $\{C\}_j$, o como las particiones $\{S\}_i$ y $\{S\}_j$) se verá completamente corrompida.

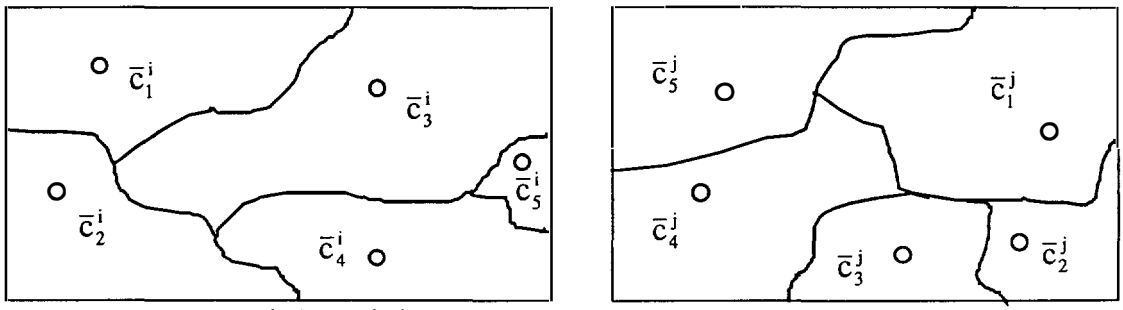

Fig.4.3 Libros de código $\{C\}_i$ y $\{C\}_j$ junto a sus respectivas particiones. Se observa la inexistencia de ordenación espacial de los vectores con relación a sus índices. Este cruce puede resultar pernicioso si no se emparejan los vectores código de $\{C\}_i$ y $\{C\}_j$ próximos entre sí.

Para formar un nuevo vector código a partir de $\{C\}_i$ y $\{C\}_j$, preservando algunas de sus características espaciales, puede pensarse en componer una correspondencia según la cual, a cada vector código de $\{C\}_i$ se le asocie con el vector código de $\{C\}_j$ más cercano a él. Formalmente, el vector de $\{C\}_j$ asociado a $\bar{c}_p^j$ será ahora:

$$as\left(\bar{c}_p^i, \{C\}_j\right) = \bar{c}_s^j \quad \text{sii} \quad \left\|\bar{c}_p^i - \bar{c}_s^j\right\|_2 \le \left\|\bar{c}_p^i - \bar{c}_q^j\right\|_2 \quad \forall\ q=0,1\ L\text{-}1. \qquad \textbf{(4.2.1.2.6)}$$

Ahora sí que tiene sentido llevar a cabo el cruce aleatorio explicado arriba con la pareja $\bar{c}_p^i$ y $\bar{c}_s^j$, pues el libro de códigos resultante mantendrá cierta similitud con ambos padres. En el caso del ejemplo anterior:

$$as\left(\bar{c}_1^i, \{C\}_j\right) = \bar{c}_5^j$$
$$as\left(\bar{c}_2^i, \{C\}_j\right) = \bar{c}_4^j$$
$$as\left(\bar{c}_3^i, \{C\}_j\right) = \bar{c}_1^j$$
$$as\left(\bar{c}_4^i, \{C\}_j\right) = \bar{c}_3^j$$
$$as\left(\bar{c}_5^i, \{C\}_j\right) \quad \bar{c}_1^j \qquad \textbf{(4.2.1.2.7)}$$

Para hacer que este operador esté equilibrado con respecto de los dos padres, cada vez que haya un cruce, se generarán dos hijos:

$$ra\left(\bar{c}^i, as\left(\bar{c}^i, \{C\}_j\right)\right)$$
$$ra\left(\bar{c}^j, as\left(\bar{c}^j, \{C\}_i\right)\right) \qquad \textbf{(4.2.1.2.8)}$$

El Cruce Constreñido que se acaba de explicar guarda una estrecha, aunque quizás velada, semejanza con el "cruce análogo" propuesto por Davidor para el diseño de trayectorias para el brazo articulado de un robot [Davidor 89].

c- Cruce Laxo:

Es semejante al anterior, pero con constricciones más laxas en cuanto a la elección de los vectores código que van a cruzarse. Específicamente, cuando se va a cruzar $\{C\}_i$ con $\{C\}_j$, se permitirá el cruce entre cada uno de los vectores $\bar{c}_r^i$ de $\{C\}_i$, y uno de los vectores $\bar{c}_{r1}^j$, $\bar{c}_{r2}^j$, , $\bar{c}_{rS}^j$, (los S vectores código de $\{C\}_j$ más cercanos a $\bar{c}_r^i$).

Debe notarse que cuando S=1 el cruce laxo es equivalente al cruce constreñido.

4.2.1.3- Mutación

Con una probabilidad P_m dada, y de forma independiente, se determina si se mutan o no cada una de las M componentes de los L vectores código de las librerías $\{C\}_i$ y $\{C\}_j$ sobre las que se aplica el operardor. La mutación sobre $c_{p,k}^i$, esto es sobre el componente k del vector p-ésimo de la librería $\{C\}_i$, se establecerá de la siguiente manera.

Sea $mm(i,k)$ la media muestral de la componente k-ésima de la librería $\{C\}_i$ a lo largo de sus L vectores código:

$$mm(i,k) \quad \frac{1}{L}\sum_{r=0}^{L-1} c'_{r,k} \qquad \textbf{(4.2.1.3.1)}$$

Sea $dm(i,k)$ la desviación muestral de la componente k-ésima de la librería $\{C\}_i$ a lo largo de sus L vectores código, tomada como:

$$dm(i,k) = \frac{1}{L}\sum_{r=0}^{L-1}\left(c_{r,k}^{i} - mm(i,k)\right)^{2} \qquad \textbf{(4.2.1.3.2)}$$

y sea $rg(\mu,\sigma^2)$ un valor al azar tomado de una distribución gaussiana de media μ y varianza σ^2.

Según esto, el nuevo valor de la componente $c_{p,k}^{i}$ será:

$$c_{p,k}^{i} = rg(mm(i,k), dm((i,k)) \qquad \textbf{(4.2.1.3.3)}$$

esto es, un valor aleatorio tomado de una distribución gaussiana de media, la media muestral entre las componentes k de los vectores código de $\{C\}_i$, y de varianza, su varianza muestral.

Este operador tenderá a infligir perturbaciones a los vectores código, llevándolos hacia posiciones céntricas en la nube de vectores código de una librería, como se muestra en la Figura 4.4.

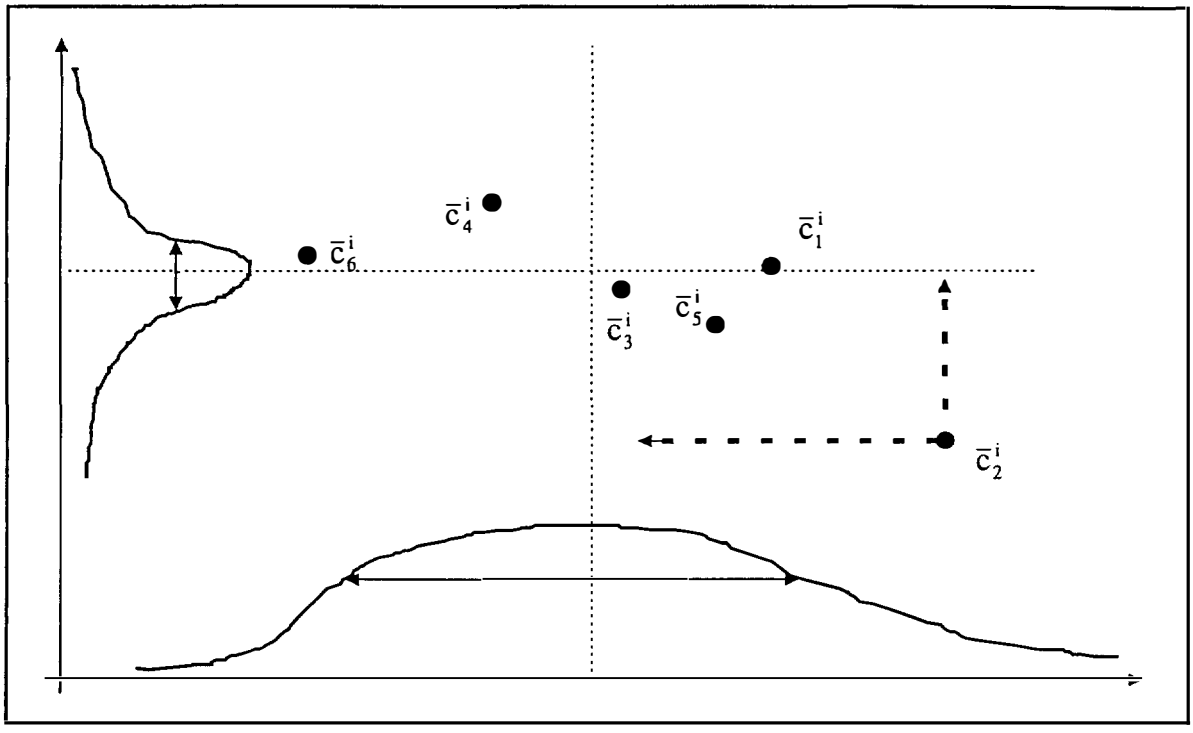

Fig. 4.4 Se dibujan seis vectores código de la librería $\{C\}_i$ y las supuestas funciones de distribuciones, medias y varianzas muestrales de cada una de las componentes de estos vectores (se asume dimensión 2, para mayor claridad). Las flechas de trazos muestran la tendencia al desplazamiento de cada una de las componentes del vector $\bar{c}_2^i$, en el caso de que fueran mutadas.

4.2.1.4- Estrategias de evolución

En el diseño de CV, el algoritmo GLA converge hacia valores de distorsión monótonamente decrecientes, por lo que resulta conveniente tomarlo como procedimiento de optimización local en la búsqueda genética.

Nuestro AGCV incorpora las estrategias Baldwiniana y Lamarckiana (selecionables externamente) descritas en 2.2.10, en las cuales un número ξ de iteraciones del GLA son tomadas como operador de optimización local. Este número se denominará "factor de aceleración GLA" y podrá ser fraccionario: por ejemplo si ξ=1/3, se aplicará una iteración básica GLA cada tres generaciones del genético.

4.2.1.5- Tratamiento de clones

También el AGCV incorpora las dos técnicas para la eliminación de individuos repetidos vistas en 2.2.11: remoción de clones y prevención de incestos, permitiendo además no aplicar ninguna de ellas (opción de no actuar).

4.2.1.6- Elistismo

El AGCV incorpora el elitismo, preservando el mejor individuo de una generación y pasándolo sin cruce ni mutación a la generación siguiente (no obstante, en la estrategia Baldwiniana, este individuo sí que es optimizado por medio del algoritmo LBG).

4.2.2- Selección de opciones y ajuste de parámetros

En este apartado se describe el proceso de prueba por el que paso a paso se van seleccionando las mejores opciones del AGCV y eligiendo los valores más adecuados para sus parámetros.

La señal de prueba utilizada ha sido la imagen Lena de 256x256 pixeles con 8 bits/pixel de precisión. De ésta se han extraído 4096 vectores de entrenamiento de dimensión 16, formados por bloques de 4x4 pixeles sin solapar. El tamaño del libro de códigos buscado es de 64 vectores y se considera que el canal introduce errores en los bits con una probabilidad (BER) de 0.01. Todo ello constituye un escenario intermedio entre los conjuntos más extensos de prueba, con los que los métodos propuestos en esa Tesis se contrastarán con otros recogidos en la literatura especializada.

La figura de mérito contemplada ha sido el coste de cuantificación o distorsión media del proceso, expresada en (2.1.4.6.4) y en (4.1.3). Sin embargo, para comparar de forma "justa", desde el punto de vista computacional, distintas configuraciones del algoritmo o distintos algoritmos, es preciso que los tiempos de cómputo empleados en todas ellas sean semejantes. Para ello se introduce el concepto de "número de iteraciones equivalentes del GLA" ("nieg"), definido como el cociente entre el coste computacional de una determinada ejecución del AG (o de cualquier otro algoritmo de diseño de VQ) y el de una iteración del algoritmo GLA, para las mismas circunstancias (imagen, PEB, tamaño del libro de código y dimensiones de los vectores). En el Apéndice 1 se dan expresiones explícitas aproximadas calculadas analíticamente del "nieg" en el AGCV y en los demás algoritmos contrastados en esta Tesis.

Destacamos cuatro conjuntos de ensayos diferentes, con los que de un modo gradual se ha tratado de ir llegando a optimizar las características del AG, a la vez que se estudiaba la influencia de las distintas opciones probadas.

a- El primer conjunto de ensayos ha ido encaminado a seleccionar una estrategia de selección, un tipo de operador de cruce y un operador para el tratamiento de los clones entre los descritos, que ofrezca buenas prestaciones para tomarlo como base para las siguientes optimizaciones. Esta etapa de pruebas no ha sido muy exhaustiva, por el motivo de ser sólo preliminar y no contener las alternativas más críticas del genético.

Para todos los casos considerados se ha fijado una probabilidad de cruce de 0.75, una probabilidad de mutación de 0.01, un tamaño de población de 15 y un factor de aceleración GLA de 2.

De los $3^{3=}27$ casos posibles (combinaciones de las tres estrategias de selección, las tres formas de cruzamiento y los tres tratamientos de clones descritas) sólo se han considerado los 7 que se recogen en la Tabla 1.

En cada uno de estos casos, el AGCV es lanzado 15 veces independientes entre sí, registrándose la distorsión media y el número iteraciones equivalentes. Las medias y desviaciones estándar de estas figuras de mérito se muestran en la Tabla 4.1.

	Distorsión media	Número de generaciones	'Num. iteraciones equivalentes de GLA'
Selección: **P** Cruce: **C** Tratam. clones: **R**	0.01918 (2.3 E-4)	71.3 (18)	3161 (823)
Selección: **R** Cruce: **C** Tratam. clones: **R**	0.01915 (1.5 E-4)	75.9 (18)	3367 (788)
Selección: **T** Cruce: **C** Tratam. clones: **R**	0.01911 (2.2 E-4)	76.3 (17)	3388 (799)
Selección: **P** Cruce: **L** Tratam. clones: **R**	0.01919 (2.0 E-4)	71.3 (17)	3161 (793)
Selección: **P** Cruce: **A** Tratam. clones: **R**	0.01926 (2.3 E-4)	59.3 (19)	2627 (850)
Selección: **P** Cruce: **C** Tratam. clones: **P**	0.01911 (2.3 E-4)	81.3 (22)	3624 (982)
Selección: **P** Cruce: **C** Tratam. clones: **N**	0.01924 (3.0 E-4)	79.0 (17)	3510 (762)

Tabla 4.1 Resultados del primer conjunto de pruebas. Valores medios, sin paréntesis; desviaciones estándar, entre paréntesis. Debe tenerse en cuenta la siguiente nomenclatura: Selección (**P**=aleatoria Proporcional, **R**= por Rango, **T**= por Torneo), Cruce (**A**=Aleatorio, **C**=Constreñido, **L**=Laxo), Tratamiento de clones (**R**: Remoción, **P**=Prevención de incestos, **N**=No actuar).

Podemos observar la escasa incidencia de la elección de unas u otras opciones. En cualquier caso, elegimos la opción (Selección **T**, Cruce **C** y Tratamiento de clones **R**) para las siguientes pruebas.

b- El siguiente conjunto de pruebas pretende determinar cuál estrategia de evolución es la más idónea. Sin embargo, las prestaciones de ambas podrían estar condicionadas por

el tamaño de la población del genético. Es por ello que se consideran conjuntamente ambos factores (tipo de estrategia L ó B y tamaño de la población de 5, 15, 25 ó 35 individuos), según se recoge en la Tabla 4.2. Las probabilidades de cruce y mutación, y el factor de aceleración GLA se mantienen igual que en las pruebas anteriores en los valores de 0.75, 0.01 y 2, respectivamente. En esta segunda colección de ensayos, en cada caso se efectuarán 15 repeticiones independientes del AG y se calculan la media muestral y la desviación estándar de la distorsión y del "nieg".

	Estrategia Lamarckiana			Estrategia Baldwiniana		
Tamaño de Población	Distorsión media	Núm. de generac.	"nieg"	Distorsión media	Núm. de generac.	"nieg"
5	0.0196 (3.7 E-4)	67.3 (18)	994 (266)	0.0237 (5.8 E-4)	39.7 (16)	582.5 (243.8)
15	0.01911 (2.2 E-4)	76.3 (17)	3388 (799)	0.0229 (4.1 E-4)	37.7 (16)	1658 (715)
25	0.01900 (2.0 E-4)	74.5 (19)	5510 (1444)	0.0229 (2.9 E-4)	41.7 (25)	3062 (1900)
35	0.01896 (1.6 E-4)	78.8 (19)	8163 (1983)	-	-	-

Tabla 4.2 Resultados del segundo conjunto de pruebas (valores medios, sin paréntesis desviaciones estándar, entre paréntesis).

Se puede observar que la estrategia Lamarckiana es siempre superior a la Baldwiniana, por lo que se elegirá para las optimizaciones subsiguientes.

c- A continuación se trata de hallar la población y el factor de aceleración GLA que dan óptimas prestaciones al AGCV. Obviamente cuanto mayor sean los valores que tomen estos parámetros, mejores librerías de código se encontrarán, pero también serán mayores las necesidades computacionales. Es, por tanto, preciso que las comparaciones se efectúen sobre ensayos con el mismo coste computacional. Por ello, para cada caso considerado en este apartado, se llevan a cabo sucesivas ejecuciones del AGCV hasta que la suma total de iteraciones equivalentes llegue a las 50.000. Seguidamente estos ensayos se agruparán en bloques que engloban varias ejecuciones del AGCV, llegando conjuntamente cada bloque a superar las 10.000 iteraciones equivalentes. De las distorsiones medias de cada bloque se extrae el mínimo (coste mínimo de bloque o "cmb"). Luego se calcula la media muestral y valor mínimo del "cmb" a lo largo de los distintos bloques.

En la Tabla 4.3 se muestran estas figuras de mérito para los casos considerados. Estos son las combinaciones posibles de poblaciones de tamaño 11, 15, 21, 25 y 31; y "factores de aceleración GLA" de valor 1/4, 1/2, 1, 2, 4, 6 y 8.

Se puede apreciar una ligera disminución de la distorsión al incrementarse el número de elementos de la población hasta llegar a los 25 individuos, donde esta tendencia se disipa o incluso se invierte en algunos casos. En cuanto al "factor de aceleración GLA", cuando aumenta su valor, las prestaciones del algoritmo mejoran claramente hasta llegar alrededor de 4 y 6, puntos en los que esta tendencia comienza a invertirse. Teniendo en cuenta estos resultados, se seleccionarán para la última etapa de

optimización del AGCV un tamaño de población de 25 y un "factor de aceleración GLA" de 5.

Factor de aceleración GLA	Tamaño de población = 11		Tamaño de población = 15		Tamaño de población = 21		Tamaño de población = 25		Tamaño de población = 31	
	med	mín	med	min	med	min	med	min	med	min
1/4	2.044	1.983	2.077	2.022	2.048	1.977	2.047	1.978	2.034	1.992
1/2	1.991	1.965	1.982	1.945	1.973	1.945	1.959	1.934	1.968	1.932
1	1.946	1.911	1.935	1.918	1.922	1.902	1.916	1.89	1.929	1.904
2	1.904	1.853	1.899	1.88	1.901	1.88	1.893	1.869	1.902	1.860
4	1.885	1.848	1.898	1.872	1.895	1.859	1.877	1.835	1.872	1.852
6	1.897	1.883	1.858	1.852	1.880	1.870	1.870	1.861	1.879	1.847
8	1.899	1.858	1.893	1.858	1.898	1.876	1.876	1.865	1.894	1.887

Tabla 4.3. Resultados multiplicados por 100 del tercer conjunto de pruebas (valores medios y mínimos de los "cmb".

d- Finalmente se pretende analizar el comportamiento del algoritmo ante distintos valores de las probabilidades de cruce y mutación. Como las influencias que ejercen estos parámetros suelen estar interrelacionadas [Murata 96], se han estudiado conjuntamente. Todas las combinaciones del producto cartesiano de las probabilidades de cruce 1.0, 0.75, 0.5, 0.25 y 0.0, con las probabilidades de mutación 0.1, 0.01, 0.001 y 0.0 han sido probadas.

El resto de parámetros seleccionados han sido las opciones vencedoras en los conjuntos de pruebas anteriores: Selección por Torneo, Cruce Constreñido, Remoción de Clones, Estrategia Lamarckiana, poblaciones de 25 individuos y "factor de aceleración GLA" de 5. Los resultados se recogen en la Tabla 4.4. y corresponden a mediciones del mismo tipo que en el ensayo anterior.

	P_c=1.0		P_c=0.75		P_c=0.5		P_c=0.25		P_c=0.0	
	med	min	med	min	med	min	med	min	med	min
P_m = 0.1	1.923	1.909	1.925	1.910	1.928	1.898	1.923	1.911	1.924	1.903
P_m = 0.01	1.888	1.860	1.883	1.865	1.889	1.862	1.892	1.875	1.869	1.848
P_m = 0.001	1.927	1.894	1.919	1.906	1.928	1.891	1.935	1.897	1.921	1.894
P_m = 0.0	1.931	1.887	2.002	1.960	1.981	1.944	1.959	1.910	1.948	1.911

Tabla 4.4. Resultados multiplicados por 100 del cuarto conjunto de pruebas (valores medios y mínimos de los "cmb".

Se observa un claro mínimo a lo largo de la variable probabilidad de mutación, en torno a 0.01. En el caso de la probabilidad de cruce no se observan tendencias claras, pero se seleccionará el valor 0.0, para el que la distorsión alcanza un valor mínimo cuando la probabilidad de mutación es de 0.01.

Es interesante este resultado por cuanto indica la escasa eficacia del operador Cruce Constreñido. Es de suponer que los otros dos tipos de cruce, descartados en el primer bloque de pruebas, tampoco mejorarán las prestaciones del genético, por lo que, a partir de ahora, ningún tipo de cruce será incorporado al mismo.

4.3- Algoritmo heurístico para la optimización conjunta de vectores e índices

En este apartado, se propone un algoritmo heurístico para el diseño de cuantificadores vectoriales en canales ruidosos basado en la repetición de ejecuciones del GLA, el cual incorpora dos importantes aspectos que abajo se comentan.

Por un parte, dado su carácter local, el algoritmo GLA se suele ejecutar repetidas veces, partiendo de librerías de códigos diferentes e independientes entre sí en cada ejecución. De todas ellas se elige la librería que da lugar a una distorsión menor. Por tanto, la única información que se va propagando a lo largo de las distintas ejecuciones es la distorsión mínima obtenida hasta al momento y la librería con la que se ha obtenido esa distorsión mínima. Más aún, esa información se almacena, pero no se utiliza en el transcurso de las ejecuciones del GLA, puesto que cada una de ellas es independiente de todas las demás, por el hecho de que el punto de partida es una librería de códigos hallada de forma completamente aleatoria e independiente del resto del proceso.

Sin embargo, la información desplegada en cada ejecución GLA es abundante, pues son evaluadas varias librerías. Incluso sin apartarse demasiado del método clásico de las repeticiones del GLA, y solamente tomando en consideración la mejor de las librerías evaluadas hasta el momento, parece razonable aprovechar, de alguna manera, la información relacionada con esta librería a la hora de confeccionar las librerías iniciales en las siguientes ejecuciones del GLA.

En segundo lugar, conviene reparar en la estrategia de optimización global, según la cual, al principio la inspección se hace muy global y poco específica, tratándose de abarcar lo más posible el espacio de búsqueda, sin perseguir que las soluciones lleguen a ser siquiera localmente óptimas. Progresivamente la búsqueda se hace más y más localizada; la exploración del espacio va dejando paso a la explotación de las buenas soluciones encontradas hasta el momento. Las técnicas "termoestadísticas" y los algoritmos "fuzzy" incorporan de forma explícita este modo de operación. Muchas otras técnicas, incluidos los Algoritmos Genéticos, particularmente aquéllos que se hibridan con otros optimizadores locales, también siguen esta pauta de forma más o menos implícita.

El algoritmo propuesto en esta sección, denominado Algoritmo Refinado de Lloyd (ARL), también presenta una progresiva "depuración" o "refinado" de la solución, conforme a la anterior estrategia de búsqueda.

4.3.1- Descripción

Cuando se realiza una ejecución del GLA, la distorsión total del sistema está dada por la ecuación 2.1.4.6.4. Suponiendo además una asignación binaria natural B_{Π_N} (ver apartado (2.1.4.1)), esta distorsión sólo será función explícita de la librería de códigos y podrá ponerse como:

$$D = D(\{C\}) = \frac{1}{N}\sum_{i=0}^{L-1}\sum_{j=0}^{L-1}\sum_{k=0}^{N_i-1} P_{ji}\left\|\bar{v}_{ik} \quad \bar{c}_j\right\|_2 \qquad \textbf{(4.3.1.1)}$$

Se puede extraer la parte de la distorsión correspondiente a la partición V_i como:

$$D_i = \frac{1}{N}\sum_{j=0}^{L-1}\sum_{k=0}^{N_i-1} P_{ji}\left\|\bar{v}_{ik} - \bar{c}_j\right\|_2 \qquad , 0 \le i \le L-1 \qquad \textbf{(4.3.1.2)}$$

De esta forma, la distorsión total puede entenderse como suma de estas distorsiones parciales:

$$D = \sum_{i=0}^{L-1} D_i \qquad \textbf{(4.3.1.3)}$$

En el método propuesto, de acuerdo con lo explicado con anterioridad, se realizan diversas ejecuciones del GLA. Para confeccionar el libro de códigos inicial, en cada una de ellas se preservan los mejores vectores de la mejor librería encontrada hasta el momento, a los cuales se añaden otros vectores tomados al azar entre el conjunto de entrenamiento. El criterio para determinar la bondad de los vectores código es precisamente su distorsión parcial: el vector $\bar{c}_i$ se considerará tanto mejor cuanto menor sea D_i.

El algoritmo completo se dibuja en la Figura 4.5 y se describe a continuación. Después de elegir aleatoriamente un libro de códigos inicial, se llevan a cabo R ejecuciones del algoritmo GLA, guardando siempre el mejor libro de códigos hasta el memento $\{C\}_{best}$ y su distorsión D_{min}. En todas las ejecuciones el código inicial se forma tomando los N-P mejores vectores de $\{C\}_{best}$ y P vectores de entrenamiento elegidos aleatoriamente (obviamente, al comienzo del algoritmo, todos los N vectores se eligen aleatoriamente del conjunto de entrenamiento).

Esto constituye el funcionamiento del bucle más interno del algoritmo (Paso 2), el cual es repetido sucesivas veces hasta que la distorsión global deje de disminuir ($D_{min} \ge D_1$). Cuando esto suceda, el parámetro P (número de vectores que van a ser elegidos aleatoriamente) es decrementado según una ley exponencial o lineal, que más tarde se explicará. El algoritmo acaba cuando P alcanza el valor P_{end} fijado externamente.

Considerando que el GLA da origen a librerías con distorsiones monótonamente decrecientes podemos ver el proceso entero como un procedimiento de minimización global de $D(\{C\})$. Cada vez que se llama al GLA, hay un movimiento en el espacio de los vectores código, que converge a una posición localmente mínima de $D(\{C\})$. Tras esto, se regeneran P vectores de $\{C\}$ y el GLA es invocado una vez más hacia un nuevo punto localmente mínimo. Mientras P es alto, una zona relativamente amplia del espacio de búsqueda es inspeccionada; conforme P decrece, la búsqueda se circunscribe en zonas más localizadas. A este proceso de "refinado" progresivo de las soluciones es al que se refiere el nombre que se ha dado al método.

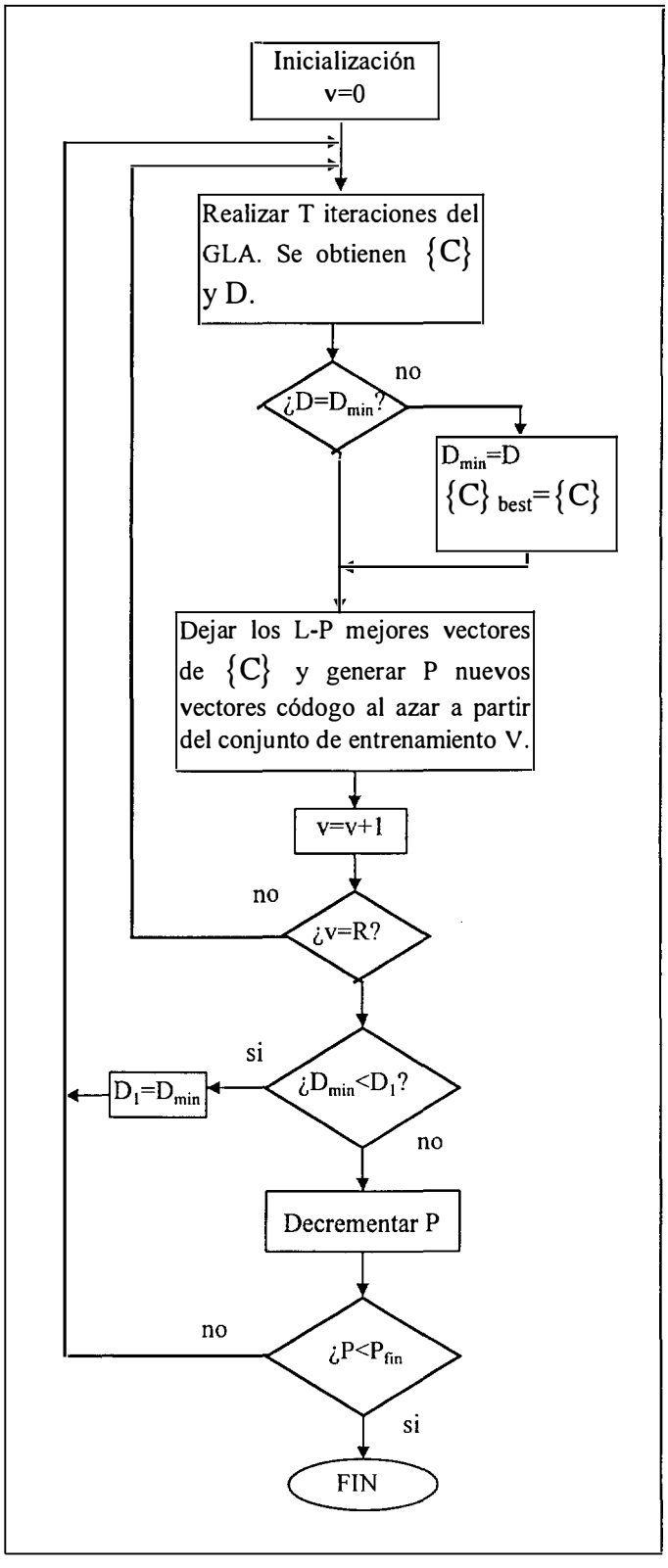

Fig. 4.5.a Algoritmo Refinado de Lloyd (diagrama de flujo).

Paso 1. Inicializar.

Paso 2. REPETIR R veces:
2.1- Ejecutar T iteraciones simples del GLA obteniendo el libro de códigos $\{C\}$ y la distorsión D.
2.2- SI $D<D_{min}$
$D_{min}=D$
$\{C\}_{best}=\{C\}$.
2.3- Generar P vectores código nuevos, preservando los mejores L-P vectores de la librería $\{C\}_{best}$.

Paso 3. SI $D_{min}<D_1$
3.1- $D_1=D_{min}$.
SI NO
3.2- Decrementar el parámetro P.
3.3- SI $P< P_{fin}$
PARAR

Paso 4. VOLVER al Paso 2.

Fig. 4.5.b Algoritmo Refinado de Lloyd (pseudocódigo).

4.3.2- Selección de parámetros

Para encontrar valores adecuados de los parámetros T y R, así como para elegir la ley de decrecimiento (LD) que gobierne la evolución del número P de vectores código nuevos generados en cada inicialización, se ha procedido a diversas simulaciones del algoritmo.

Se ha elegido el mismo escenario de prueba que el utilizado para seleccionar el conjunto de parámetros del AGCV: imagen blanco y negro de Lena, Lena de 256x256 pixeles con 8 bits/pixel de precisión, 4096 vectores de entrenamiento de dimensión 16, formados por bloques de 4x4 pixeles sin solapar, libros de código de 64 vectores y BER de 0.01.

Se han considerado los siguientes parámetros y leyes de decrecimiento:

$$T = \{2,\ 4,\ 6,\ 8,\ 10,\ 12\}$$
$$R = \{5,\ 10,\ 15,\ 20,\ 25,\ 30\}$$
$$LD = \{\text{Exponencial}, \text{Lineal}\}$$

Con la Ley Exponencial, el parámetro P sigue una secuencia de valores en los que cada uno es mitad del anterior. En nuestro caso, el valor inicial es 32 y el final, 2, por lo que la secuencia es $\{32 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2\}$.

Con la Ley Lineal, en el Paso 3.2 del algoritmo, P se decrementa una cantidad constante Δ. En nuestro caso, $\Delta=8$ y los valores iniciales y finales de P son, respectivamente, 34 y 2. La secuencia de valores de P es, por tanto, $\{34 \rightarrow 26 \rightarrow 18 \rightarrow 10 \rightarrow 2\}$.

Para cada configuración (tripleta T-R-LD) el ARL se ha ejecutado repetidamente hasta llegar a las 20.000 iteraciones GLA simples.

Después se calculan las distorsiones medias de cada ejecución según (4.3.1.1) y se agrupan en 5 bloques de aproximadamente el mismo tamaño. Se extrae la distorsión mínima dentro de cada bloque (distorsión "intra-bloque") y finalmente se obtiene la media y valor mínimo de estas 5 distorsiones "intra-bloque". En las Tablas 4.5 y 4.6 aparecen estos últimos estadísticos para cada uno de los valores probados de los parámetros T y R y las dos leyes de decrecimiento. Más abajo, las Figuras 4.6 y 4.7 muestran en gráficas 3-D estos resultados.

R	T=2		T=4		T=6		T=8		T=10		T=12	
	med	min	med	min	med	min	med	min	med	min	med	min
5	1,975	1,956	1,952	1.924	1,952	1,939	1.941	1.920	1,938	1.925	1,942	1.930
10	1,970	1.955	1,931	1.898	1,939	1.903	1.922	1.901	1,933	1.909	1,942	1.915
15	1,955	1.936	1,951	1.925	1,921	1.897	1.931	1.900	1,927	1.916	1,935	1.920
20	1,973	1.957	1,928	1.907	1,930	1.911	1.943	1.910	1,927	1.908	1,952	1.928
25	1,957	1.926	1,925	1.900	1,916	1.889	1.952	1.899	1,951	1.916	1,953	1.895
30	1,974	1.939	1,929	1.917	1,954	1.910	1.947	1.898	1,959	1.889	1,954	1.895

Tabla 4.5 Valores medio y mínimo (escalados por 100) de la distorsión "intra-bloque". (Ley exponencial)

R	T=2		T=4		T=6		T=8		T=10		T=12	
	med	min	med	min	med	min	med	min	med	min	med	min
5	1,975	1.958	1,946	1.922	1,931	1.928	1,930	1.911	1,933	1.906	1,932	1.891
10	1,955	1.943	1,930	1.906	1,927	1.897	1,937	1.916	1,924	1.893	1,925	1.902
15	1,947	1.911	1,925	1.902	1,922	1.891	1,929	1.900	1,921	1.884	1,918	1.901
20	1,962	1.924	1,925	1.909	1,928	1.906	1,933	1.909	1,941	1.886	1,910	1.892
25	1,968	1.934	1,933	1.887	1,917	1.899	1,934	1.910	1,950	1.900	1,930	1.900
30	1,978	1.958	1,923	1.901	1,938	1.899	1,948	1.901	1,922	1.903	1,931	1.900

Tabla 4.6 Valores medio y mínimo (escalados por 100) de la distorsión "intra-bloque". (Ley lineal)

Ya que 26 de los 36 casos considerados han dado menores distorsiones para la Ley Lineal que para la exponencial, ésta es la que elegiremos para el algoritmo. El siguiente paso, es el de elegir un buen par de parámetros R y T. Podemos ver en la Tabla 4.5 y en la Figura 4.6 que conforme T asciende de 2 a 6, la distorsión se reduce. A partir de este punto, la tendencia se hace indefinida sin apreciarse mejora ni empeoramiento en las prestaciones.

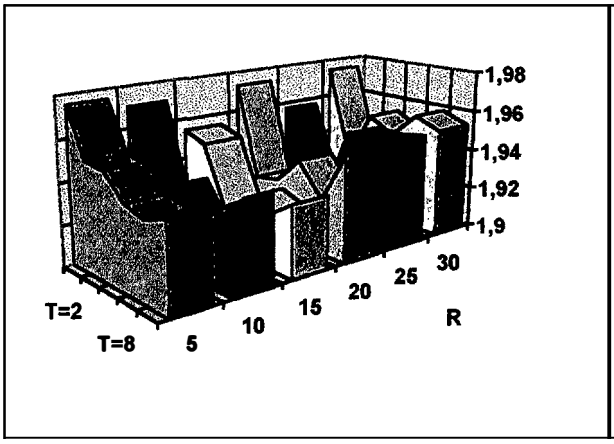

Fig. 4.6 Valores medios de distorsiones "intra-bloque". (Ley exponencial).

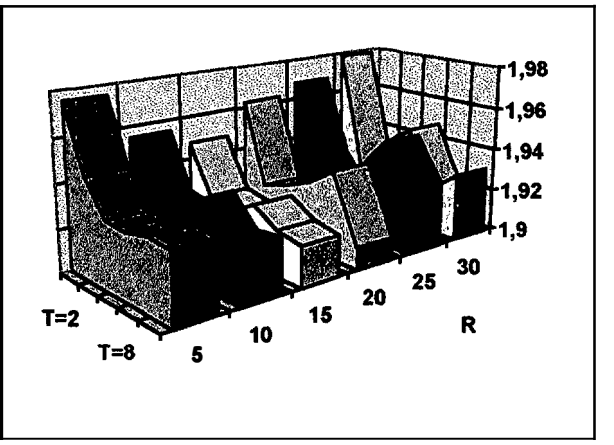

Fig 4.7. Valores medios de distorsiones "intra-bloque". (Ley lineal).

Tampoco los resultados son muy esclarecedores con relación a R. No obstante, se puede atisbar un ligero patrón cóncavo: la distorsión disminuye conforme R aumenta hasta un cierto valor, desde el cual, comienza a crecer. El punto mínimo, donde se invierte la tendencia, es distinto en cada caso, pero se podría situar aproximadamente en R=15. El juego de parámetros elegido es, por tanto, (T=6, R=15, Ley Lineal).

4.4- Optimización alternada de vectores e índices basada en Algoritmos Genéticos

En esta sección se propone un nuevo método para el diseño de cuantificadores vectoriales en condiciones de canal con ruido, el cual lleva a cabo una búsqueda alternada e iterativa de la librería de códigos y de la asignación de índices. Este método hace uso del algoritmo GLA para optimizar la librería y de un AG especialmente diseñado para buscar la asignación de índices. Por ello recibe el nombre de Genético Híbrido para la Cuantificación Vectorial (GHCV).

En la presente Tesis no se ha realizado un estudio exhaustivo para determinar las mejores opciones del GHCV en cuanto a operadores genéticos empleados, valor de los parámetros, etc. Tampoco ha sido sometido a las pruebas comparativas de simulación con las que los otros dos métodos aquí presentados (AGCV y ARL) han sido evaluados y contrastados, y que se recogen en el capítulo 5.

El algoritmo GHCV se presenta, por tanto, únicamente como una posible vía alternativa, aún sin explorar a fondo, para el diseño de cuantificadores vectoriales, basado en Algoritmos Genéticos. En [Malanda 98] este algoritmo se describe someramente y se detallan los resultados de simulaciones realizadas para comprimir señales artificiales que atienden a una distribución Gauss-Markov de primer orden. Las menores distorsiones obtenidas en comparación con el algoritmo GLA dan muestra de sus posibilidades.

4.4.1- Planteamiento

Tal y como se expresaba en la Introducción de este capítulo, siempre que esté presente la RGVMP, la distorsión global del sistema depende de dos de sus variables, el libro de códigos y la función de asignación:

$$D = D(\{C\}, \Pi) \qquad \textbf{(4.4.1.1)}$$

Además, según lo explicado en las secciones 2.1.4.4, 2.1.4.5 y 2.1.4.6, la aplicación alternada de las reglas RGVMP y RGC en el algoritmo GLA van "moviendo" el libro de códigos $\{C\}$ hacia posiciones de distorsión localmente mínima, sin cambio alguno de la función de asignación.

Por otra parte, los AG pueden utilizarse para optimizar la asignación de índices Π partiendo de un libro de códigos dado, el cual no será modificado en el proceso. Tal es el caso del algoritmo presentado en [Pan 96].

Sin embargo, los procesos de optimización de $\{C\}$ y de Π no son independientes, como se puede intuir de la estrecha relación funcional entre $\bar{c}_i$ y P_{ji} en la función de distorsión dada en (2.1.4.6.4), que se repite abajo para mayor claridad:

$$D = \frac{1}{N}\sum_{i=0}^{L-1}\sum_{k=0}^{N_i-1}\sum_{i=0}^{L-1} P_{ji}\left\|\bar{v}_{ij} - \bar{c}_i\right\|_2 \qquad \textbf{(4.4.1.2)}$$

Más aún, la RGC dada en (2.1.4.6.7) refleja la dependencia explícita entre los vectores código óptimos y las probabilidades P_{ji}:

$$\bar{c}_i = \frac{\sum_{j=0}^{L-1} P_{ji} \cdot P_j \cdot \hat{c}_j}{\sum_{j=0}^{L-1} P_{ji} \cdot P_j} \qquad i=0, 1, \ L-1 \qquad \textbf{(4.4.1.3)}$$

estando P_{ji} relacionada con la función de asignación Π a través de (2.1.4.3.4).

Aun no teniendo una expresión analítica implícita ni explícita para la asignación óptima de un sistema de CV con errores de canal, es evidente que esta asignación es fuertemente dependiente del lugar que ocupen en el espacio los vectores de la librería de códigos.

Teniendo presente estas consideraciones, se puede concluir que la optimización independiente de $\{C\}$ y de Π, optimizando primero el libro de códigos y después la asignación, no conducirá, en general, a soluciones de distorsión globalmente mínima.

Por el contrario, el método GHCV explicado a continuación, plantea el proceso de búsqueda de forma alternada e iterativa: un paso para optimizar $\{C\}$ seguido otro para optimizar Π, repitiéndose esta operación hasta finalizar la búsqueda.

4.4.2- Descripción

Según lo expuesto más arriba, el GHCV contiene dos bloques fundamentales:

- un GLA que lleva a cabo la búsqueda del libro de códigos óptimo a partir de una asignación fija,

- un AG que trata de optimizar la asignación de índices para una librería de códigos dada. A este bloque se le llamará Asignador Genético de Indices (AGI).

El funcionamiento del algoritmo GHCV se muestra en la Figura 4.8. Tras la inicialización, se suceden T iteraciones del GLA y S generaciones del bloque AGI. Ambas constituyen el bucle principal del algoritmo, el cual se repite hasta que se cumpla algún criterio de convergencia, por ejemplo, que el descenso relativo de la distorsión en las últimas iteraciones deje de superar cierto valor umbral.

El bloque AGI es un Algoritmo Genético en el que los individuos son asignaciones tentativas

$$B_{\Pi_i} = \{\Pi_i(0), \Pi_i(1), \ , \Pi_i(L-1)\} \qquad \textbf{(4.4.2.1)}$$

donde i se refiere al elemento i-ésimo de la población del genético. Los individuos son, por tanto, permutaciones posibles de la lista correspondiente a la asignación natural Π_N:

$$B_{\Pi_N} = \{0, 1, \ , L-1\} \qquad \textbf{(4.4.2.2)}$$

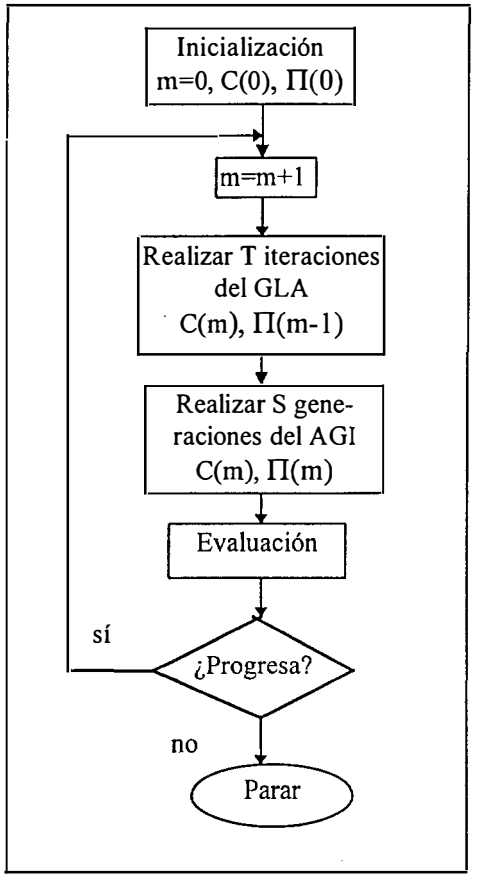

Fig. 4.8.a Algoritmo GHCV (diagrama de flujo)

Paso 1. Inicialización del libro de códigos
Paso 2. Realizar T iteraciones del GLA
Paso 3. Realizar S iteraciones del AGI
Paso 4. SI la distorsión se reduce,
saltar al Paso 2
Paso 5. SI NO,
parar

Fig. 4.8.b Algoritmo GHCV (pseudocódigo)

La función de coste es nuevamente la distorsión media dada en (2.1.4.6.4). Como a lo largo de las S iteraciones del bloque AGI, la librería de códigos se mantiene fija, esta distorsión será función sólo de Π:

$$D = D(\Pi) \qquad \textbf{(4.4.2.3)}$$

En general, podrían incluirse muy diversas alternativas en cuanto a los criterios de selección, mecanismos de mutación y cruce, optimizadores locales y tratamiento de clones, aunque aquí no se detalla ninguna de ellas en particular, por no haberse realizado estudios comparativos, como ya se comentó anteriormente.

Capítulo 5

EXPERIMENTOS

5.1- Algoritmos sometidos al estudio comparativo

En esta Tesis se han hecho numerosas simulaciones, que más adelante se relatarán, para comparar los siguientes algoritmos:

a- $AGCV_1$:

El algoritmo AGCV explicado en la sección 4.2. y optimizado con el conjunto de parámetros y opciones obtenidas a partir de las pruebas preliminares allí comentadas: Selección por Torneo, no utilización de cruces, Remoción de Clones, Estrategia Lamarckiana, población del genético de 25 individuos, "factor de aceleración GLA" de 5 y probabilidad de mutación de 0.01.

b- $AGCV_2$:

El algoritmo AGCV explicado en la sección 4.2, pero no optimizado. Se incluye con el propósito de comprobar la robustez de este método genético. El conjunto de opciones seleccionadas es la siguiente: Selección por Rango, Cruce Constreñido, Prevención de Incestos, Estrategia Lamarckiana, población del genético de 15 individuos, "factor de aceleración GLA" de 2, probabilidad de cruce de 0.6 y probabilidad de mutación de 0.01.

c- ARL:

El algoritmo explicado en la sección 4.3, con el juego de parámetros óptimos allí obtenido: T=6, R=15, Ley Lineal.

d- TD:

El Temple Determinístico [Miller 94], descrito en la sección 3.3.2.1. La evolución del parámetro β, no especificada en la referencia anterior, se realiza mediante una ley exponencial dada por:

$$\beta[n] = \alpha \cdot \beta[n \quad 1] \qquad \textbf{(5.1.1)}$$

donde n indica la iteración del proceso.

Tras probar con diferentes valores de $\beta[1]$, y de α, se eligieron 1 y 1.08, respectivamente. Por otra parte, no siendo el TD un algoritmo que haga decrecer la

distorsión siempre de forma monótona, es preciso establecer un número máximo de iteraciones, que, en este caso, se fijó en 2000.

e- GLA:

Según se describía en la sección 2.1.4.5, con el diseño basado en datos empíricos (sección 2.1.4.6).

f- AIW:

La Asignación de Indices de Wu y Barba [Wu 93], comentada en la sección 3.3.1.1.b.

5.2- Banco de pruebas

Para probar y contrastar los algoritmos anteriores se han usado tres conocidas imágenes (Lena, Baboon y Pepper), cada una de 256x256 pixeles, con 8 bits/pixel de precisión. De nuevo se han tomado como conjunto de entrenamiento, 4096 vectores de dimensión 16, correspondientes a bloques no solapados de 4x4 pixeles.

Se han probado libros de código de tamaño 32, 64 y 128, correspondientes a 0.312, 0.375 y 0.437 bits/pixel, respectivamente. Por otra parte, se ha considerado que el canal es simétrico y que presenta una probabilidad de error por bit de 1.0E-1, 1.0E-2, 1.0E-3 ó 1.0E-4.

5.3- Cálculo de la distorsión

En esta sección se describe una forma explícita y muy eficiente para calcular la distorsión global del sistema dada en (2.1.4.6.4) La derivación es similar a la utilizada en [Secker 92] para el diseño de codificadores Trellis, pero extrapolada aquí para el caso de CV.

Cuando está presente la RGVMP, la partición está completamente determinada por el libro de códigos $\{C\}$ y la distorsión sólo depende de $\{C\}$ y de Π. En el Apéndice 1 se muestra que esta distorsión puede expresarse como:

$$D(\{C\},\Pi) \quad \frac{1}{N}\sum_{s=0}^{N-1}\sum_{d=1}^{M} v_s^2(d) + \frac{1}{N}\sum_{s=0}^{N-1} \min_{i\in\{0,\ ,L-1\}} \left\{\sum_{d=1}^{M}\left(\beta_{id} - \alpha_{id} v_s(d)\right)\right\} \qquad \textbf{(5.3.1)}$$

donde $v_s(d)$ y $c_j(d)$ son la componente d-ésima de los vectores $\vec{v}_s$ y $\vec{c}_j$, respectivamente. Por su parte,

$$\alpha_{id} = 2\sum_{j=0}^{L-1} a_{ijd} \qquad \textbf{(5.3.2)}$$

$$\beta_{id} = \sum_{j=0}^{L-1} a_{ijd} c_j(d) \qquad \textbf{(5.3.3)}$$

siendo

$$a_{ijd} \quad Prob(j / i)\, c_j(d), \qquad \textbf{(5.3.4)}$$

El primer término es la energía media de los vectores en el conjunto de entrenamiento, el cual será constante a lo largo del proceso, por lo que sólo será necesario calcularlo una sola vez, o incluso ignorarlo, pues no aportará información diferencial al contrastar diferentes libros de código.

Los coeficientes α_{id} y β_{id} son independientes del vector analizado (variable s). Así que sólo requerirán ser calculados a lo largo de las variables i y d, pero, de nuevo, no a lo largo de s. Por consiguiente, en el bucle más interno de (5.3.1), únicamente hay que calcular un producto y una diferencia. Este siempre constituye la parte de mayor coste computacional en los algoritmos de diseño de CV para canales ruidosos, en particular en los métodos AGCV, ARL, DA y GLA, que más tarde se contrastarán.

5.4- Figuras de Mérito

La figura de mérito empleada es la llamada Relación Señal de Pico a Ruido (PSNR), calculada como [Zeger 92]:

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{255^2}{\mathrm{D/M}}\right) \qquad \textbf{(5.4.1)}$$

Sin embargo, todos los algoritmos anteriores excepto el AIW van obteniendo las solución de forma iterativa, por lo que en cada iteración la PSNR alcanzada es diferente, pudiéndose expresar como PSNR[n]. En este sentido hay que tener en cuenta que normalmente, no sólo es importante la PSNR obtenida al final del proceso, sino también la que se va obteniendo a lo largo del mismo. Según esto se proponen las siguientes figuras de mérito:

a- PSNR integral:

$$\mathrm{PSNR}_g = \frac{1}{N_2 - N_1 + 1}\sum_{n=N_1}^{N_2}\mathrm{PSNR}[n] \qquad \textbf{(5.4.2)}$$

que resulta una media de los valores que va tomando la PSNR entre N_1 y N_2, iteraciones inicial y final que van a ser consideradas. En todos los ensayos efectuados en este trabajo, N_2 toma el valor de la iteración final y $N_1 = 0.15 \cdot N_2$ (la razón por la cual no hacer que N_1 tome el valor 1 es evitar que los estados transitorios del algoritmo lleguen a afectar de manera apreciable esta medida).

b- PSNR final:

$$\mathrm{PSNR}_f = \mathrm{PSNR}[N_2] \qquad \textbf{(5.4.3)}$$

Para cada tamaño de la librería de códigos, y del BER probados, AGCV_1 fue repetido 5 veces, guardando la distorsión media a lo largo de las iteraciones y el número total de iteraciones equivalentes del GLA ("nieg").

AGCV_2, ARL y TD fueron procesadas diversas veces hasta que el "nieg" hubo llegado al valor obtenido en las 5 ejecuciones del AGCV_1. En cada uno de los procesos

se recogieron las distorsiones integral y final y se llevó a cabo el siguiente procedimiento de evaluación:

Paso 1. Calcular el PSNR de cada ejecución r e iteración n, llamándolo PSNR[r,n].

Paso 2. REPETIR PARA p=1 HASTA 5

2.1- Agrupar varias ejecuciones consecutivas del algoritmo de tal forma que en cada grupo el "nieg" total alcance el "nieg" de la ejecución p de $AGCV_1$.

2.2- Llamar $PSNR[r_p,n]$ al PSNR correspondiente a estas ejecuciones, con $r_p=1,2,\ R_p$ donde R_p es el número de ejecuciones en el grupo p.

2.3- Calcular

$$PSNR^{(p)}[n] = \min_{r_p}\left\{PSNR\left[r_p, n\right]\right\}$$

2.4- Extraer $PSNR_g^{(p)}$ y $PSNR_f^{(p)}$ de $PSNR^{(p)}$ usando (5.4.2) y (5.4.3).

Paso 3. Obtener el valor medio de $PSNR_g^{(p)}$ y $PSNR_f^{(p)}$ según:

$$PSNR_g = \frac{1}{5}\sum_{p=1}^{5} PSNR_g^{(p)}$$

$$PSNR_f = \frac{1}{5}\sum_{p=1}^{5} PSNR_f^{(p)}$$

Fig. 5.1 Proceso para calcular el PSNR integral y final de $AGCV_2$ y TD.

El GLA también ha sido ejecutado varias veces hasta que el "nieg" llega a superar el valor obtenido en las 5 ejecuciones del $AGCV_1$. Dado que el número de ejecuciones del GLA es elevado, para medir las dos PSNR se utiliza un procedimiento muy estable, basado en el análisis estadístico. Para su cálculo se ha derivado una regla analítica para la esperanza de la distorsión mínima en función del número de ejecuciones GLA inspeccionadas.

En el Apéndice 3 se obtiene la función EC(N,P), que expresa la distorsión mínima obtenida en P ejecuciones del GLA seleccionadas aleatoriamente entre un conjunto total de N ejecuciones. El descenso de EC en función de P es equivalente a las funciones de coste de AGCV o TS en función de sus correspondientes costes computacionales, dados en términos de su "nieg".

Para obtener figuras de mérito coherentes se procede según se indica en la Figura 5.2.

Paso 1. Calcular PSNR[n] como

$$PSNR[n] = 10 \log_{10} \left(\frac{255^2}{\frac{E(N,n)}{M}} \right)$$

Paso 2. REPETIR PARA p=1 HASTA 5

2.1- Sea N_2 el "nieg" de la ejecución p-ésima de $AGCV_1$.

2.2- Sea N_1=0.15 N_2

2.3- Calcular $PSNR_g^{(p)}$ y $PSNR_f^{(p)}$ según (5.4.2) y (5.4.3).

Paso 3. Obtener el valor medio de $PSNR_g^{(p)}$ y $PSNR_f^{(p)}$ según:

$$PSNR_g = \frac{1}{5} \sum_{p=1}^{5} PSNR_g^{(p)}$$

$$PSNR_f = \frac{1}{5} \sum_{p=1}^{5} PSNR_f^{(p)}$$

Fig. 5.2. Proceso para calcular el PSNR integral y final del GLA.

El algoritmo AIW no tiene en cuenta las probabilidades de error en la transmisión. Su proceso comienza a partir de un libro de códigos previamente optimizado (generalmente obtenido para el caso de transmisión sin error) y rápidamente converge, recolocando los índices de forma determinística.

La mayor parte del coste computacional de este método reside en la determinación previa del libro de códigos inicial, la cual es independiente del propio AIW. Por ello el parámetro $PSNR_g$ no tiene demasiado interés en este caso. La manera de computar el $PSNR_f$ es la siguiente:

Paso 1. REPETIR PARA p=1 HASTA 5

1.1- Obtener un buen libro de códigos inicial asumiendo que PEB=0.0., usando $AGCV_1$ o cualquier otro procedimiento.

1.2- Reasignar los índices del libro de códigos usando AIW.

1.3- Calcular

$$PSNR_f^{(p)} = 10 \log_{10} \left(\frac{255^2}{\frac{D^{(p)}}{M}} \right)$$

donde $D^{(p)}$ es la distorsión debida al libro de códigos con los índices reasignados.

Paso 2. Extraer el valor medio de esta medida como:

$$PSNR_f = \frac{1}{5} \sum_{p=1}^{5} PSNR_f^{(p)}$$

Fig. 5.3. Proceso para calcular el PSNR final del AIW.

5.5- Resultados de las simulaciones

En las Tablas 5.1-5.3 se muestran las medias y desviaciones estándar de los 5 valores de $PSNR_g^{(p)}$ y $PSNR_f^{(p)}$ (p=1, .. 5), obtenidas en las simulaciones correspondientes a las tres imágenes de prueba, los seis métodos bajo análisis, los 4 valores de BER considerados y librerías de código de tamaño 32. En las Figuras 5.4-5.7, los resultados de las tablas pueden verse en curvas superpuestas (no se incluyen la correspondiente al método AIW debido a que aparecería mucho más abajo, haciendo disminuir la aparente diferencia entre el resto de las curvas. También se dejan de incluir, por el mismo motivo, los puntos correspondientes al BER=1.0E-1).

	BER=1.0E-4		BER=1.0E-3		BER=1.0E-2		BER=1.0E-1	
	$PSNR_g$	$PSNR_f$	$PSNR_g$	$PSNR_f$	$PSNR_g$	$PSNR_f$	$PSNR_g$	$PSNR_f$
$AGCV_1$	30.39 (8.7E-3)	30.41 (1.1E-2)	29.85 (1.3E-2)	29.88 (2.0E-2)	28.09 (2.4E-2)	28.12 (2.5E-2)	21.89 (4.7E-4)	21.89 (3.4E-4)
$AGCV_2$	30.33 (7.4E-2)	30.35 (7.2E-2)	29.82 (2.4E-2)	29.84 (2.8E-2)	28.12 (1.9E-2)	28.15 (3.6E-2)	21.89 (1.5E-4)	21.89 (0.0)
ARL	30.36	30.38	29.81	29.84	28.13	28.15	21.89	21.89
TD	27.29 (1.6E-1)	27.41 (1.8E-1)	27.87 (1.6E-1)	28.59 (5.0E-2)	27.19 (1.2E-1)	27.91 (1.1E-1)	21.50 (7.4E-3)	21.88 (2E-10)
GLA	30.12 (3.0E-2)	30.17 (6.9E-3)	29.67 (1.1E-2)	29.72 (2.9E-3)	27.96 (2.2E-2)	28.00 (5.4E-3)	21.83 (2.6E-2)	21.89 (3.2E-6)
AIW	-	30.17 (4.7E-2)	-	28.10 (3.4E-1)	-	21.38 (7.1E-1)	-	12.50 (6.9E-1)

Tabla 5.1 Medias y desviaciones estándar (entre paréntesis) de $PSNR_g$ y $PSNR_f$ de la CV de la imagen Lena con libros de código de tamaño L=32.

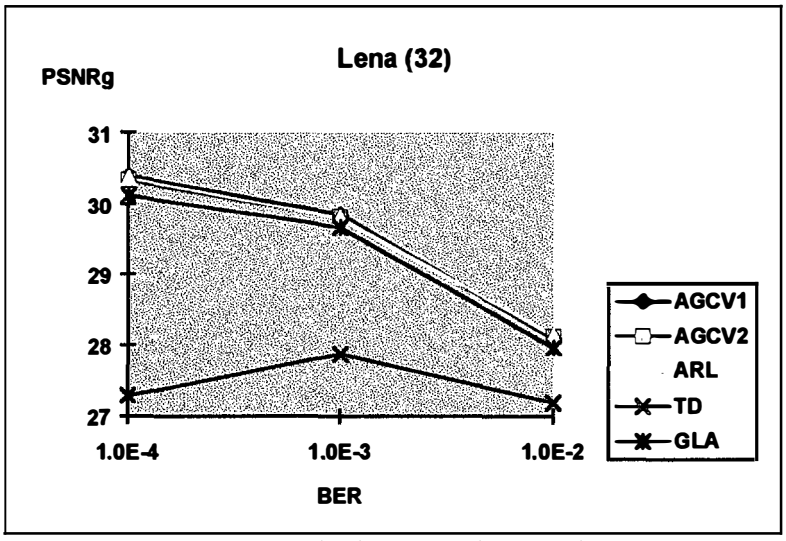

Figure 5.4.(a) Media del $PSNR_g$ de la CV de la imagen Lena con libros de código de tamaño L=32.

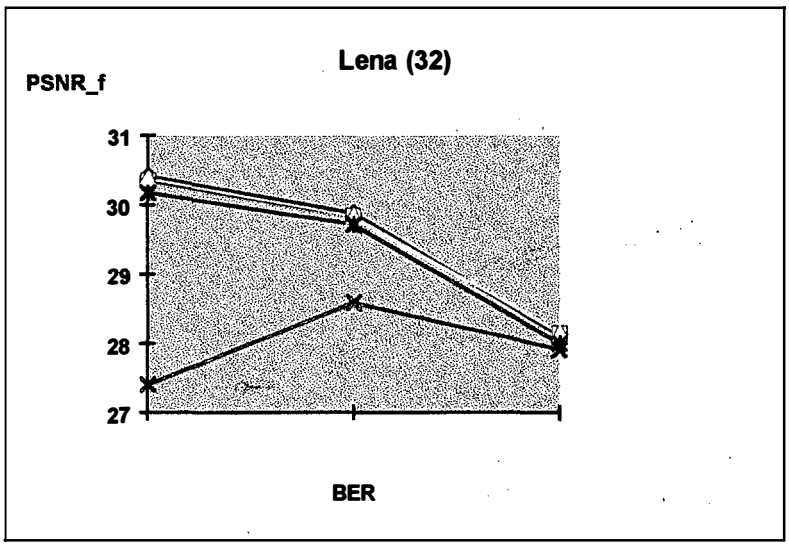

Figure 5.4.(b) Media del $PSNR_f$ de la CV de la imagen Lena con libros de código de tamaño L=32.

	BER=1.0E-4		BER=1.0E-3		BER=1.0E-2		BER=1.0E-1	
	$PSNR_g$	$PSNR_f$	$PSNR_g$	$PSNR_f$	$PSNR_g$	$PSNR_f$	$PSNR_g$	$PSNR_f$
$AGCV_1$	23.06 (3.1E-3)	23.07 (3.7E-3)	22.96 (1.1E-2)	22.96 (1.2E-2)	22.48 (1.5E-2)	22.49 (1.5E-2)	20.28 (6.6E-4)	20.30 (5.9E-4)
$AGCV_2$	23.06 (7.3E-3)	23.06 (4.8E-3)	22.95 (1.3E-2)	22.95 (1.3E-2)	22.47 (1.3E-2)	22.49 (8.7E-3)	20.30 (7.1E-4)	20.30 (5.6E-4)
ARL	23.04	23.06	22.95	22.98	22.49	22.51	20.30	20.30
TD	22.64 (4.1E-2)	23.06 (6.3E-3)	22.29 (2.3E-1)	22.94 (6.5E-2)	22.22 (1.6E-1)	22.50 (2.3E-3)	19.97 (1.5E-2)	20.15 (7.3E-5)
GLA	22.99 (1.5E-2)	23.04 (7.6E-4)	22.88 (2.1E-2)	22.92 (7.7E-4)	22.35 (1.7E-2)	22.42 (1.5E-3)	20.24 (3.1E-2)	20.30 (9.2E-5)
AIW	-	23.07 (4.3E-3)	-	22.92 (1.6E-2)	-	21.67 (1.1E-1)	-	16.83 (2.8E-1)

Tabla 5.2 Medias y desviaciones estándar (entre paréntesis) de $PSNR_g$ y $PSNR_f$ de la CV de la imagen Baboon con libros de código de tamaño L=32.

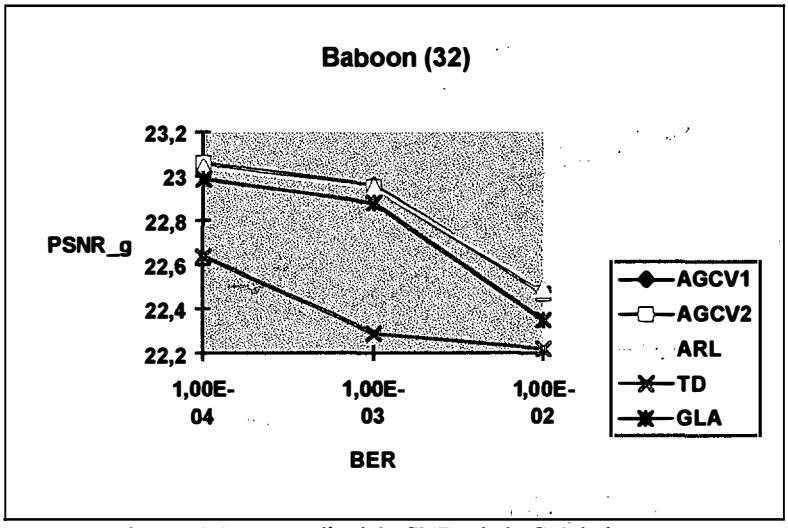

Figure 5.5.(a) Media del $PSNR_g$ de la CV de la imagen Baboon con libros de código de tamaño L=32.

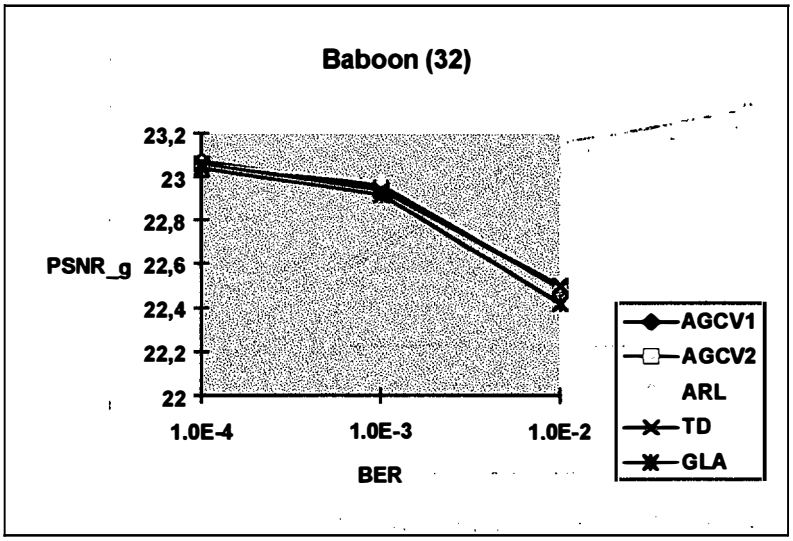

Figure 5.5.(b) Media del $PSNR_f$ de la CV de la imagen Baboon con libros de código de tamaño L=32.

	BER=1.0E-4		BER=1.0E-3		BER=1.0E-2		BER=1.0E-1	
	$PSNR_g$	$PSNR_f$	$PSNR_g$	$PSNR_f$	$PSNR_g$	$PSNR_f$	$PSNR_g$	$PSNR_f$
$AGCV_1$	26.20 (5.4E-3)	26.21 (7.6E-3)	25.87 (3.1E-2)	25.90 (3.7E-2)	24.39 (1.9E-2)	24.42 (2.3E-2)	19.83 (8.5E-4)	19.83 (5.4E-4)
$AGCV_2$	26.18 (8.3E-3)	26.19 (1.0E-2)	25.85 (1.8E-2)	25.87 (2.7E-2)	24.38 (2.7E-2)	24.43 (3.8E-2)	19.83 (2.4E-3)	19.83 (5.4E-4)
ARL	26.19	26.20	25.86	25.91	24.44	24.45	19.83	19.83
TD	25.59 (7.2E-2)	26.11 (4.2E-3)	25.38 (2.1E-1)	25.81 (2.9E-2)	24.00 (1.3E-1)	24.36 (4.2E-2)	19.41 (1.8E-2)	19.62 (1.1E-3)
GLA	26.09 (3.5E-2)	26.16 (2.9E-3)	25.67 (1.8E-2)	25.72 (1.7E-4)	24.19 (6.7E-3)	24.28 (8.8E-3)	19.80 (1.9E-2)	19.83 (1.6E-5)
AIW	-	26.18 (8.5E-3)	-	25.48 (3.7E-2)	-	21.59 (1.3E-1)	-	13.99 (1.5E-1)

Tabla 5.3 Medias y desviaciones estándar (entre paréntesis) de $PSNR_g$ y $PSNR_f$ de la CV de la imagen Pepper con libros de código de tamaño L=32.

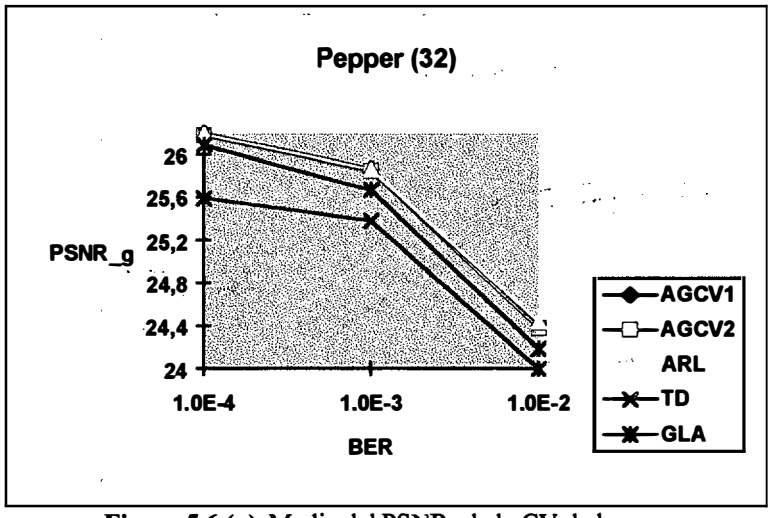

Figure 5.6.(a) Media del $PSNR_g$ de la CV de la imagen Pepper con libros de código de tamaño L=32.

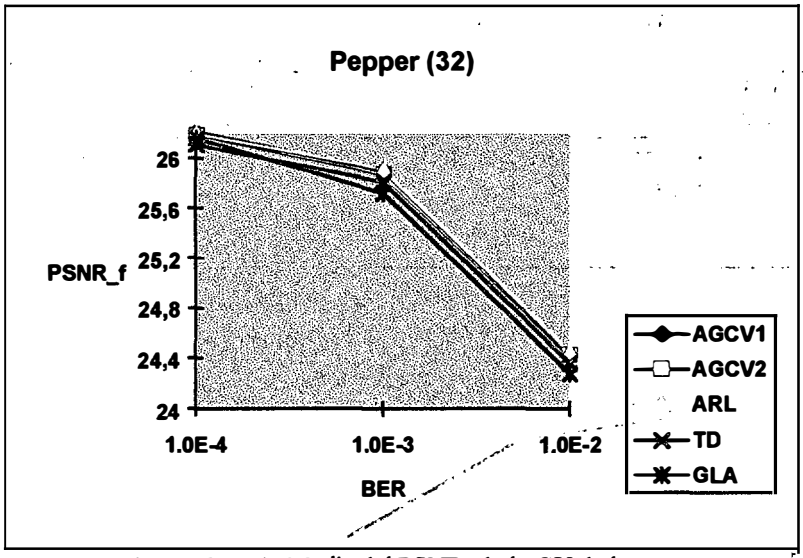

Figure 5.6.(b) Media del $PSNR_f$ de la CV de la imagen Pepper con libros de código de tamaño L=32.

El resultado que más resalta es la extrema degradación que presenta AIW para valores altos del BER. Aparte de esto, se puede observar que los dos genéticos y el ARL obtienen menores distorsiones que sus competidores en todos los casos estudiados, no existiendo diferencias apreciables entre ellos.

Es muy sorprendente lo sensible que resulta TD a la imagen particular que se está cuantificando: para el caso de Lena y un BER de 1.0E-4, los valores de la PSNR integral apareen 3 dB por debajo de los obtenidos con los otros algoritmos. Cuando el BER disminuye a 1.0E-3, 1.0E-2 y 1.0E-1, esta diferencia disminuye a 2 dB, 1 dB y 0.3 dB aproximadamente. Sin embargo, para las imágenes Baboon y Pepper, este PSNR está solamente entre 0.25 y 0.65 dB por debajo de los genéticos y de ARL (de nuevo dicha diferencia disminuye a medida que el BER aumenta). Se pueden extraer consideraciones similares en relación al PSNR final obtenido con TD: la distorsión en Lena es inaceptablemente alta, aunque en Pepper resulta casi tan buena como en los genéticos o en ARL (sólo 0.1 ó 0.2 dB peor) y en Baboon la diferencia es virtualmente despreciable.

Por otra parte, $AGCV_1$, $AGCV_2$ y ARL superan al GLA (una cantidad que varía entre 0 y 0.25 dB) en ambas relaciones PSNR, para valores de BER inferiores a 1.0E-1.

Las Tablas 5.4-5.6 son análogas a las tres previas, pero ahora corresponden a librerías de código de tamaño 64. En las figuras 5.7-5.9 se muestran en diferentes curvas los resultados de estas tablas.

	BER=1.0E-4		BER=1.0E-3		BER=1.0E-2		BER=1.0E-1	
	$PSNR_g$	$PSNR_f$	$PSNR_g$	$PSNR_f$	$PSNR_g$	$PSNR_f$	$PSNR_g$	$PSNR_f$
$AGCV_1$	31.81 (1.6E-2)	31.85 (2.6E-2)	31.31 (6.4E-2)	31.37 (7.2E-2)	29.28 (4.4E-2)	29.31 (5.5E-2)	22.80 (1.7E-3)	22.80 (5.0E-4)
$AGCV_2$	31.75 (3.4E-2)	31.80 (3.3E-2)	31.24 (3.5E-2)	31.28 (3.2E-2)	29.24 (2.3E-2)	29.29 (1.5E-2)	22.80 (6.5E-4)	22.80 (7.6E-5)
ARL	31.82	31.85	31.26	31.28	29.23	29.26	22.80	22.80
TD	28.12 (6.8E-2)	28.48 (6.4E-2)	28.68 (1.6E-1)	29.16 (2.2E-1)	27.72 (1.2E-1)	28.51 (2.2E-1)	22.31 (3.7E-2)	22.75 (1.9E-3)

GLA	31.48 (3.5E-2)	31.54 (6.3E-3)	31.07 (1.4E-2)	31.10 (4.8E-3)	29.04 (7.2E-3)	29.07 (0.0)	22.71 (4.6E-2)	22.79 (2.7E-4)
AIW	-	30.51 (7.5E-2)	-	28.57 (4.1E-1)	-	20.95 (7.0E-1)	-	12.12 (6.3E-1)

Tabla 5.4 Medias y desviaciones estándar (entre paréntesis) de $PSNR_g$ y $PSNR_f$ de la CV de la imagen Lena con libros de código de tamaño L=64

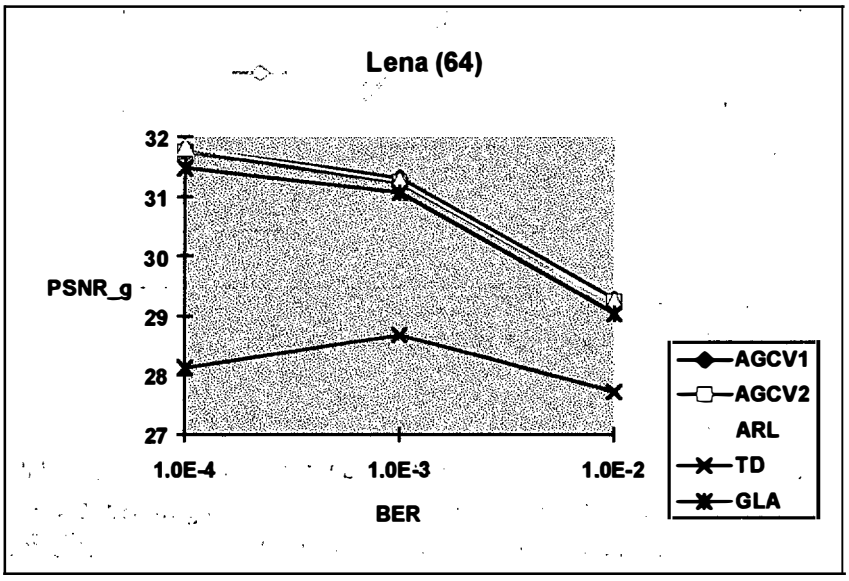

Figure 5.7.(a) Media del $PSNR_g$ de la CV de la imagen Lena con libros de código de tamaño L=64.

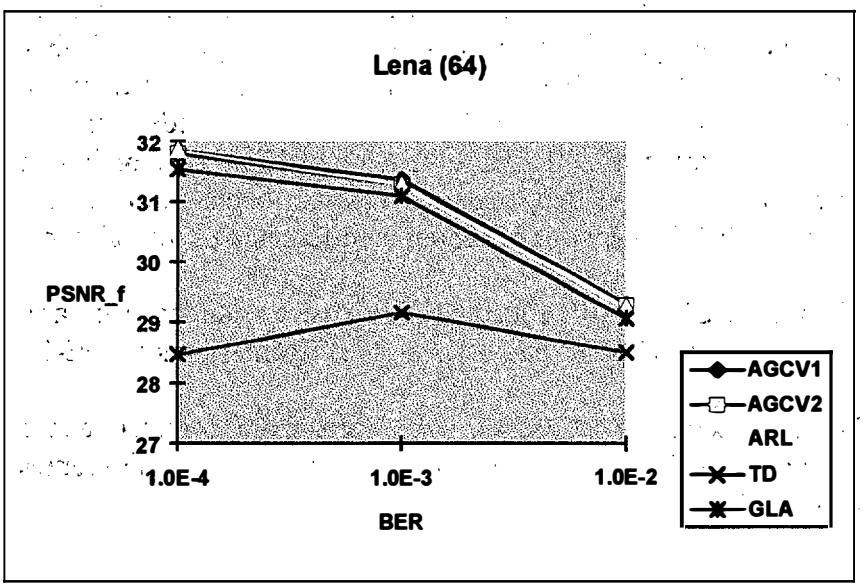

Figure 5.7.(b) Media del $PSNR_f$ de la CV de la imagen Lena con libros de código de tamaño L=64.

	BER=1.0E-4		BER=1.0E-3		BER=1.0E-2		BER=1.0E-1	
	$PSNR_g$	$PSNR_f$	$PSNR_g$	$PSNR_f$	$PSNR_g$	$PSNR_f$	$PSNR_g$	$PSNR_f$
$AGCV_1$	23.70 (4.1E-3)	23.71 (4.8E-3)	23.57 (9.4E-3)	23.58 (9.9E-3)	22.98 (1.7E-2)	23.00 (1.1E-2)	20.64 (1.0E-3)	20.64 (1.0E-3)
$AGCV_2$	23.69 (5.6E-3)	23.70 (5.8E-3)	23.55 (6.9E-3)	23.56 (1.1E-2)	22.96 (1.7E-2)	22.99 (1.6E-2)	20.64 (7.1E-4)	20.64 (6.9E-4)
ARL	23.69	23.71	23.54	23.58	23.02	23.05	20.64	20.64
TD	23.04 (5.6E-2)	23.69 (3.1E-3)	22.96 (2.4E-2)	23.61 (6.6E-3)	22.72 (6.6E-2)	23.03 (6.2E-3)	20.24 (3.0E-2)	20.50 (1.8E-3)
GLA	23.61 (2.2E-2)	23.65 (9.3E-4)	23.46 (7.9E-3)	23.49 (2.8E-4)	22.82 (8.7E-3)	22.84 (1.0E-4)	20.60 (4.6E-2)	20.64 (1.8E-4)
AIW	-	23.72 (6.6E-3)	-	23.52 (1.1E-2)	-	21.90 (5.6E-2)	-	16.53 (9.5E-2)

Tabla 5.5 Medias y desviaciones estándar (entre paréntesis) de $PSNR_g$ y $PSNR_f$ de la CV de la imagen Baboon con libros de código de tamaño L=64.

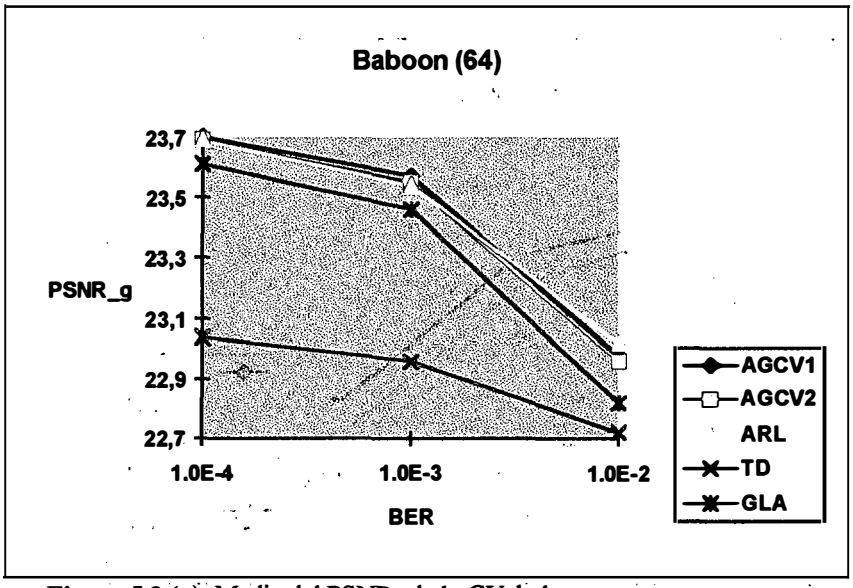

Figure 5.8.(a) Media del $PSNR_g$ de la CV de la imagen Baboon con libros de código de tamaño L=64.

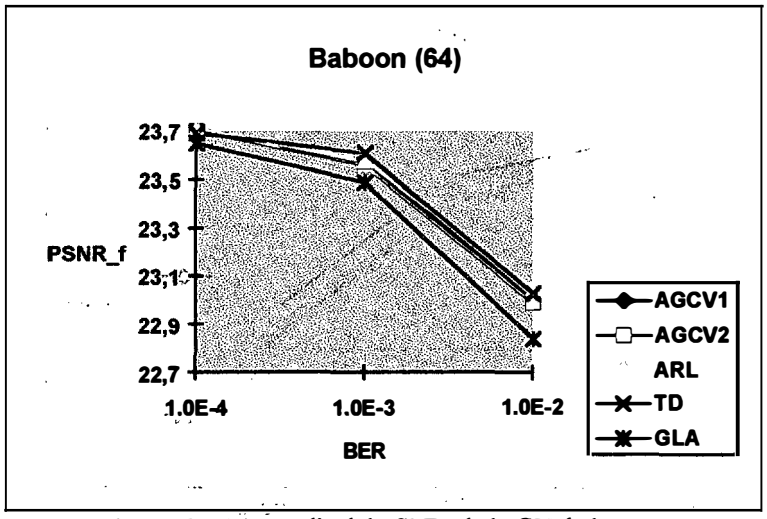

Figure 5.8.(b) Media del $PSNR_f$ de la CV de la imagen Baboon con libros de código de tamaño L=64.

	BER=1.0E-4		BER=1.0E-3		BER=1.0E-2		BER=1.0E-1	
	$PSNR_g$	$PSNR_f$	$PSNR_g$	$PSNR_f$	$PSNR_g$	$PSNR_f$	$PSNR_g$	$PSNR_f$
$AGCV_1$	27.51 (1.1E-2)	27.53 (9.8E-3)	27.07 (3.0E-2)	27.13 (2.9E-2)	25.44 (2.2E-2)	25.48 (1.4E-2)	20.43 (7.0E-3)	20.44 (6.5E-3)
$AGCV_2$	27.47 (1.6E-2)	27.49 (1.8E-2)	27.02 (3.9E-2)	27.09 (3.8E-2)	25.39 (1.9E-2)	25.45 (2.6E-2)	20.44 (2.0E-3)	20.44 (1.9E-3)
ARL	27.53	27.54	27.01	27.02	25.39	25.41	20.44	20.44
TD	26.84 (5.8E-2)	27.34 (8.2E-3)	26.27 (3.1E-1)	26.85 (2.9E-1)	24.94 (4.8E-2)	25.29 (3.9E-2)	19.94 (3.4E-2)	20.25 (1.0E-3)
GLA	27.37 (2.5E-2)	27.42 (2.2E-3)	26.75 (1.8E-2)	26.79 (1.1E-3)	25.07 (6.0E-3)	25.15 (8.1E-4)	20.40 (3.7E-2)	20.44 (1.8E-4)
AIW	-	27.49 (1.7E-2)	-	26.48 (8.7E-2)	-	21.68 (2.9E-1)	-	13.80 (2.6E-1)

Tabla 5.6 Medias y desviaciones estándar (entre paréntesis) de $PSNR_g$ y $PSNR_f$ de la CV de la imagen Pepper con libros de código de tamaño L=64.

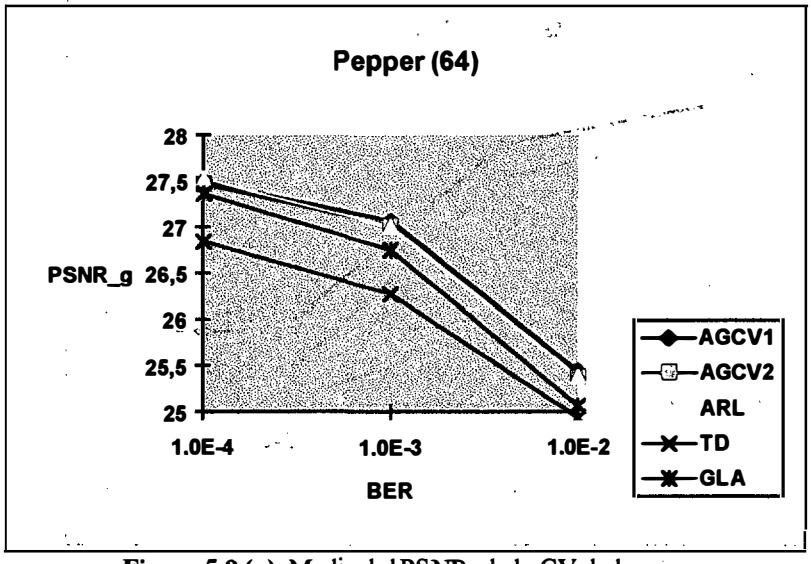

Figure 5.9.(a) Media del $PSNR_g$ de la CV de la imagen Pepper con libros de código de tamaño L=64.

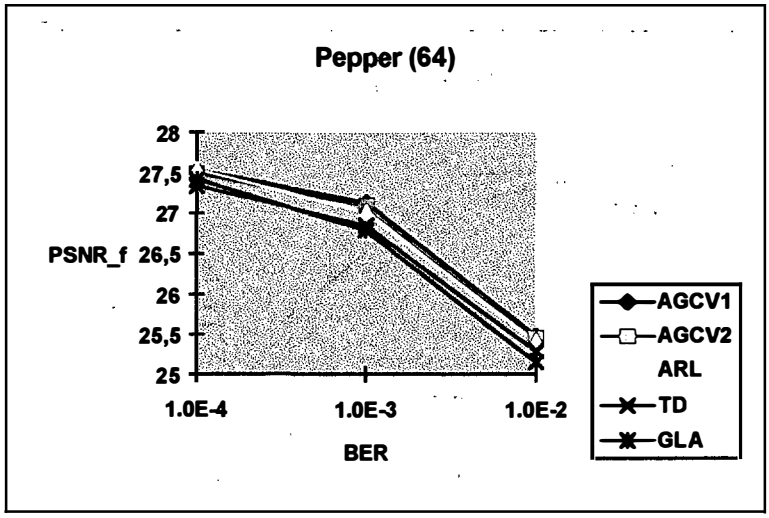

Figure 5.9.(b) Media del $PSNR_f$ de la CV de la imagen Pepper con libros de código de tamaño L=64.

En estas últimas tablas y gráficas se puede observar un comportamiento muy similar al caso previo (L=32). Nuevamente $AGCV_1$, $AGCV_2$ y ARL exhiben los mejores resultados, siendo inapreciables las diferencias entre ellos. Quizás el único nuevo aspecto relevante es la degradación ligeramente superior que presenta ahora el GLA frente a estos tres métodos (de 0 a 0.35 dB).

Los resultados correspondientes a los libros de códigos de 128 vectores se dan en las Tablas 5.7-5.9. En las Figuras 5.10-5.12 los datos de estas tablas se expresan en curvas superpuestas.

	BER=1.0E-4		BER=1.0E-3		BER=1.0E-2		BER=1.0E-1	
	$PSNR_g$	$PSNR_f$	$PSNR_g$	$PSNR_f$	$PSNR_g$	$PSNR_f$	$PSNR_g$	$PSNR_f$
$AGCV_1$	33.41 (9.9E-2)	33.51 (9.6E-2)	32.79 (4.1E-2)	32.91 (6.5E-2)	30.33 (5.8E-2)	30.40 (7.4E-2)	23.65 (1.1E-3)	23.65 (1.0E-3)
$AGCV_2$	33.39 (3.0E-2)	33.49 (3.1E-2)	32.74 (3.0E-2)	32.84 (3.4E-2)	30.33 (3.8E-2)	30.42 (1.9E-2)	23.65 (2.8E-3)	23.65 (4.8E-3)
ARL	33.70	33.74	32.89	32.90	30.17	30.19	23.65	23.66
TD	29.35 (1.7E-1)	29.71 (1.5E-1)	29.43 (1.7E-1)	29.85 (9.2E-2)	28.79 (7.8E-2)	29.20 (1.2E-1)	23.01 (5.5E-2)	23.56 (1.6E-3)
GLA	32.93 (2.3E-2)	32.98 (1.3E-2)	32.27 (1.3E-2)	32.32 (5.6E-3)	29.90 (5.7E-3)	29.94 (5.8E-3)	23.56 (4.9E-2)	23.65 (4.0E-4)
AIW	-	33.05 (1.9E-1)	-	29.32 (8.0E-1)	-	21.12 (1.2)	-	12.14 (8.5E-1)

Tabla 5.7 Medias y desviaciones estándar (entre paréntesis) de $PSNR_g$ y $PSNR_f$ de la CV de la imagen Lena con libros de código de tamaño L=128.

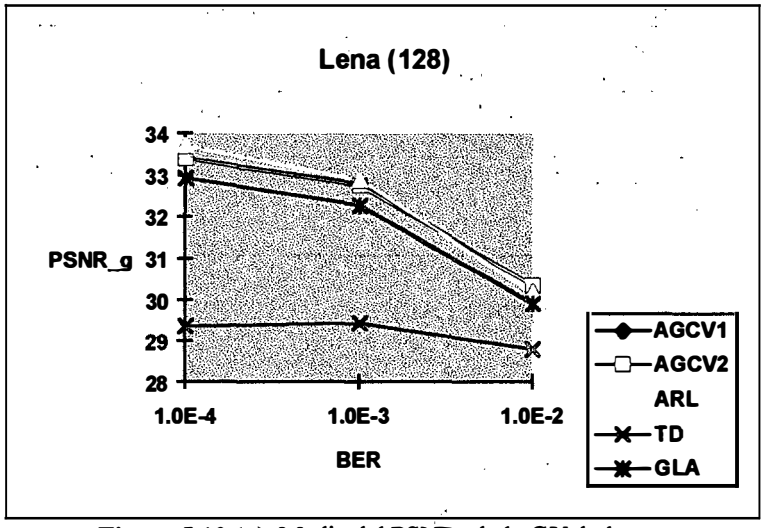

Figure 5.10.(a) Media del $PSNR_g$ de la CV de la imagen Lena con libros de código de tamaño L=128.

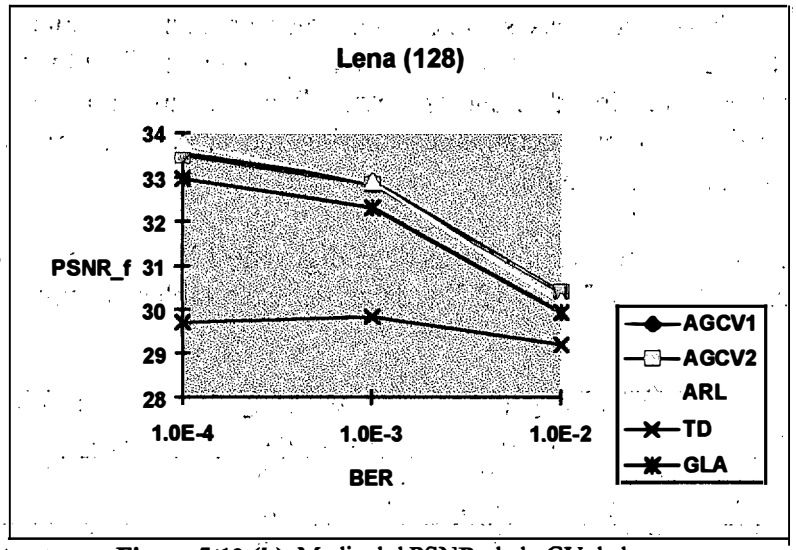

Figure 5.10.(b) Media del $PSNR_f$ de la CV de la imagen Lena con libros de código de tamaño L=128.

	BER=1.0E-4		BER=1.0E-3		BER=1.0E-2		BER=1.0E-1	
	$PSNR_g$	$PSNR_f$	$PSNR_g$	$PSNR_f$	$PSNR_g$	$PSNR_f$	$PSNR_g$	$PSNR_f$
$AGCV_1$	24.41 (5.6E-3)	24.43 (5.7E-3)	24.21 (1.5E-2)	24.24 (2.0E-2)	23.50 (1.5E-2)	23.55 (1.6E-2)	21.00 (2.5E-3)	21.00 (2.0E-3)
$AGCV_2$	24.37 (6.6E-3)	24.39 (7.1E-3)	24.21 (1.8E-2)	24.24 (2.2E-2)	23.47 (6.9E-3)	23.52 (1.4E-2)	21.00 (4.4E-3)	21.00 (4.6E-3)
ARL	24.41	24.43	24.21	24.23	23.51	23.53	21.00	21.00
TD	23.71 (5.5E-3)	24.41 (3.4E-3)	23.44 (8.5E-2)	24.30 (9.9E-3)	23.07 (1.6E-2)	23.59 (9.2E-3)	20.55 (4.4E-2)	20.85 (5.3E-3)
GLA	24.27 (9.6E-3)	24.31 (1.3E-3)	24.06 (1.8E-2)	24.08 (1.1E-3)	23.22 (0.0)	23.27 (0.0)	20.95 (1.6E-2)	20.99 (4.9E-4)
AIW	-	24.43 (1.5E-2)	-	24.14 (1.6E-2)	-	22.09 (6.2E-2)	-	16.28 (9.4E-2)

Tabla 5.8 Medias y desviaciones estándar (entre paréntesis) de $PSNR_g$ y $PSNR_f$ de la CV de la imagen Baboon con libros de código de tamaño L=128.

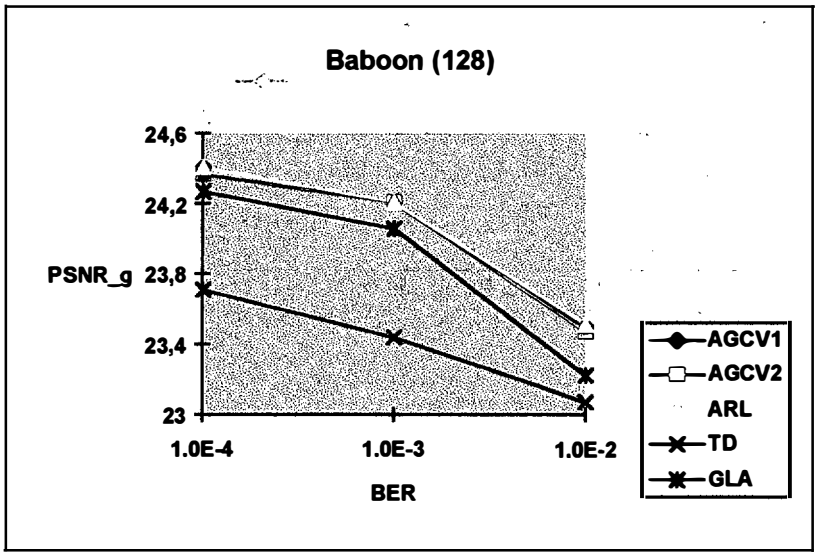

Figure 5.11.(a) Media del $PSNR_g$ de la CV de la imagen Baboon con libros de código de tamaño L=128.

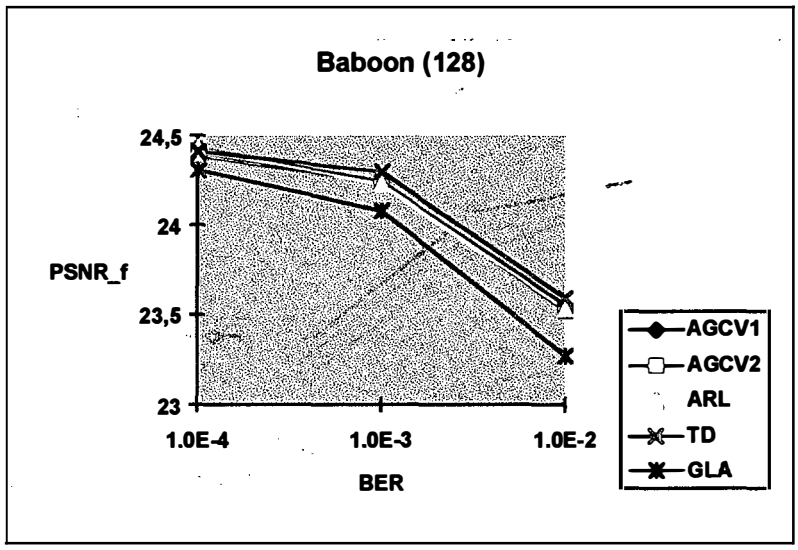

Figure 5.11.(b) Media del $PSNR_f$ de la CV de la imagen Baboon con libros de código de tamaño L=128.

	BER=1.0E-4		BER=1.0E-3		BER=1.0E-2		BER=1.0E-1	
	$PSNR_g$	$PSNR_f$	$PSNR_g$	$PSNR_f$	$PSNR_g$	$PSNR_f$	$PSNR_g$	$PSNR_f$
$AGCV_1$	28.79 (9.0E-3)	28.83 (9.0E-3)	28.24 (3.2E-2)	28.36 (4.3E-2)	26.36 (3.3E-2)	26.46 (4.2E-2)	21.06 (8.0E-3)	21.06 (8.4E-3)
$AGCV_2$	28.71 (4.3E-2)	28.73 (5.5E-2)	28.17 (1.7E-2)	28.25 (1.7E-2)	26.33 (2.3E-2)	26.40 (2.4E-2)	21.06 (3.4E-3)	21.06 (2.9E-3)
ARL	28.92	28.93	28.15	28.16	26.27	26.29	21.06	21.06
TD	27.70 (8.8E-2)	28.53 (2.6E-2)	27.14 (2.1E-1)	28.06 (5.1E-2)	25.64 (5.4E-2)	26.27 (3.2E-2)	20.55 (1.5E-1)	20.95 (1.2E-1)
GLA	28.54 (3.6E-2)	28.58 (4.3E-3)	27.72 (1.7E-2)	27.76 (2.5E-3)	25.91 (1.7E-2)	25.96 (3.0E-3)	21.00 (3.6E-2)	21.05 (3.6E-2)
AIW	-	28.76 (1.7E-2)	-	27.40 (6.1E-2)	-	21.81 (1.6E-1)	-	13.74 (1.4E-1)

Tabla 5.9 Medias y desviaciones estándar (entre paréntesis) de $PSNR_g$ y $PSNR_f$ de la CV de la imagen Pepper con libros de código de tamaño L=128.

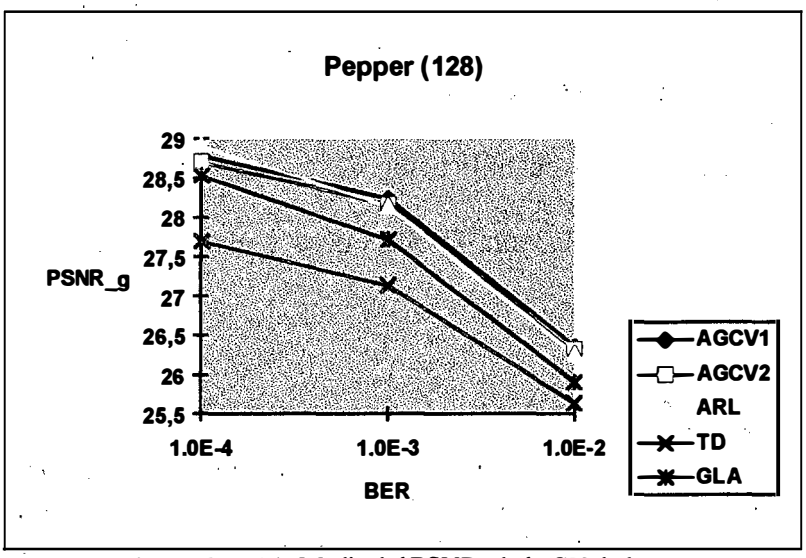

Figure 5.12.(a) Media del $PSNR_g$ de la CV de la imagen Pepper con libros de código de tamaño L=128.

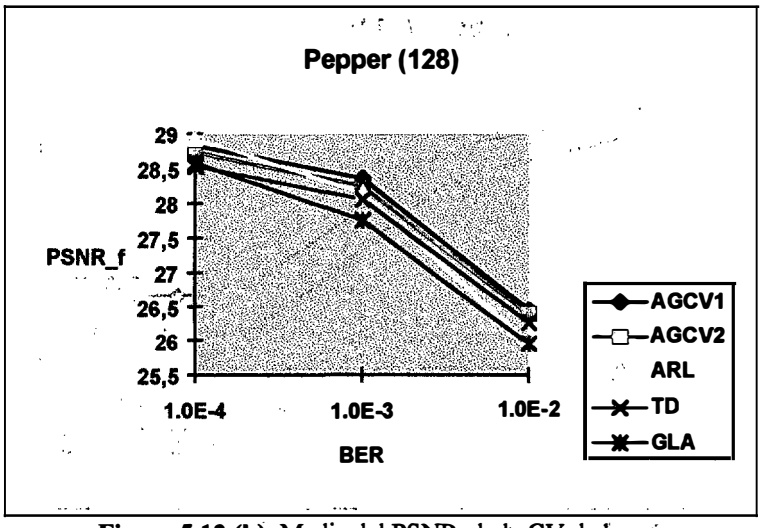

Figure 5.12.(b) Media del $PSNR_f$ de la CV de la imagen Pepper con libros de código de tamaño L=128.

Otra vez los resultados son similares a los casos L=32 y L=64. Sin embargo, puede percibirse una notable degradación en la PSNR integral de TD en las tres imágenes (no sólo en Lena), para todos los BERs estudiados. Esta degradación no aparece en la PSNR final, que exhibe un comportamiento muy similar a los casos L=32 y L=64 (en la imagen de Lena esta distorsión sigue siendo inaceptablemente alta para todos los BER considerados excepto BER=1.0E-1).

Por otra parte, las prestaciones del GLA con respecto a los métodos genéticos y al heurístico ARL son ahora peores que en los casos precedentes: de 0 a 0.6 dB, dependiendo de la imagen y el BER considerado (se excluye aquí el caso BER=1.0E-1, en el que los resultado entre los dos genéticos, ARL y GLA apenas difieren).

Por último, aunque $AGCV_1$, $AGCV_2$ y ARL sigan siendo los algoritmos de mejores prestaciones, comienza a apreciarse un comportamiento ligeramente diferente entre ellos. Por un lado, $AGCV_1$ siempre resulta superior a $AGCV_2$ aunque nunca la diferencia de sus PSNR supera los 0.11 dB. Por otra parte, el ARL resulta superior a los genéticos cuando el ruido del canal es pequeño: con una BER de 1.0E-4, obtiene entre 0.2 y 0.3 dB menos distorsión para la imagen Lena, y entre 0.1 y 0.2 para Pepper. Sin embargo, a medida que el canal se hace más ruidoso, los genéticos van superando al ARL. Así, cuando BER=1.0E-2 las PSNR resultan entre 0.15 y 0.2 dB más favorables para aquéllos que para éste, en la cuantificación de Lena y entre 0.6 y 0.17 dB más favorables, en la cuantificación de Pepper.

Para el caso de la imagen Baboon, los genéticos y el ARL llegan a resultados prácticamente idénticos.

5.6- Pruebas complementarias sobre el algoritmo ARL

Para comprobar la eficacia del proceso de refinado, en el algoritmo ARL optimizado (parámetros seleccionados según lo expuesto en el apartado 4.3.2) se ha procedido a su comparación con otro ARL en el que no se sigue una ley decreciente del número de centroides. Al contrario, este número permanece constante a lo largo del proceso, constituyendo un nuevo parámetro externo que llamaremos "ncm", acrónimo

de "número de centroides mutados". El nuevo algoritmo se puede catalogar como un ARL con Ley de Decrecimiento constante.

Este ARL ha sido sometido a diversas pruebas de simulación, con el mismo escenario y protocolo de evaluación que el descrito en el apartado 4.3.2. En todas ellas al parámetro R se de dio el valor 6, mientras que se probaron los siguientes conjuntos de valores para R y "ncm":

$$R = \{5,\ 10,\ 15,\ 20\}$$
$$\text{"ncm"} = \{5,\ 10,\ 15,\ 20,\ 25,\ 30,\ 35,\ 40,\ 45,\ 50\}$$

Los valores medios y mínimos de "distorsión intrabloque" se recogen en la Tabla 5.10. Los primeros se dibujan en la gráfica de la Figura 5.13.

"ncm"	R=5		R=10		R=15		R=20	
	med	min	med	min	med	min	med	min
5	1,9921	1,9633	1,9977	1,9523	1,9979	1,9611	2,0384	1,9738
10	1,9759	1,9521	1,9848	1,9567	1,9764	1,9621	1,9651	1,9256
15	1,9581	1,9384	1,9566	1,9405	1,9759	1,9283	1,9592	1,9333
20	1,9546	1,9364	1,9733	1,9561	1,9371	1,9274	1,9452	1,9281
25	1,9453	1,9279	1,9419	1,9177	1,9256	1,8987	1,9526	1,9228
30	1,9461	1,9295	1,9346	1,9045	1,9326	1,9146	1,9329	1,9163
35	1,9408	1,9180	1,9329	1,8994	1,9084	1,8661	1,9350	1,9149
40	1,9352	1,9283	1,9336	1,9242	1,9304	1,9048	1,9347	1,9092
45	1,9359	1,9211	1,9252	1,9149	1,9412	1,9224	1,9313	1,8993
50	1,9514	1,9285	1,9249	1,9147	1,9275	1,9083	1,9309	1,9043

Tabla 5.10. Valores medio y mínimo (escalados por 100) de la distorsión "intra-bloque" (Ley constante).

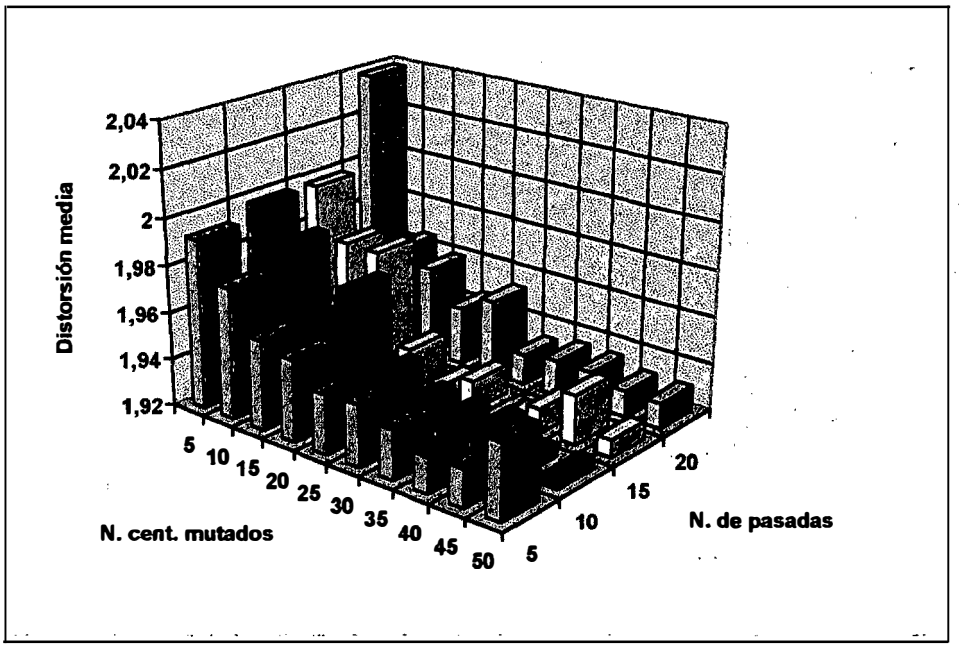

Fig. 5.13 Valores medios de la "distorsión intrabloque"

Para comparar este resultado con los obtenidos con el ARL optimizado (T=6, L=15, LD lineal) se calcula en escala logarítmica el cociente entre las distorsiones alcanzadas con ambas versiones del algoritmo, según la expresión:

$$CD = 10 \log_{10} \frac{D_{cte}(T = 6, R, \text{' ncm'})}{D_{lin}(T = 6, R = 15)} \qquad \textbf{(5.5.1)}$$

donde $D_{cte}(T, R, \text{' ncm'})$ es la distorsión del ARL con LD constante, y parámetros T, R y "ncm"; y $D_{lin}(T, R)$ es la distorsión del ARL con LD lineal.

En la Tabla 5.10 se recogen los valores medios y mínimos de CD y en la Figura 5.14 se muestra una gráfica con los valores medios de esta figura de mérito.

"ncm"	R=5		R=10		R=15		R=20	
	med	min	med	min	med	min	med	min
5	0,1556	0,1630	0,1678	0,1386	0,1682	0,1581	0,2554	0,1861
10	0,1201	0,1381	0,1396	0,1483	0,1212	0,1603	0,0963	0,0787
15	0,0808	0,1075	0,0775	0,1122	0,1201	0,0848	0,0833	0,0961
20	0,0730	0,1030	0,1144	0,1470	0,0340	0,0828	0,0521	0,0844
25	0,0523	0,0839	0,0447	0,0609	0,0081	0,0176	0,0686	0,0724
30	0,0541	0,0875	0,0284	0,0309	0,0239	0,0539	0,0246	0,0577
35	0,0423	0,0616	0,0246	0,0192	-0,0308	-0,0576	0,0293	0,0545
40	0,0297	0,0848	0,0261	0,0756	0,0189	0,0316	0,0286	0,0416
45	0,0313	0,0686	0,0072	0,0545	0,0432	0,0715	0,0210	0,0190
50	0,0659	0,0853	0,0065	0,0541	0,0124	0,0396	0,0201	0,0304

Tabla 5.10 Valores medio y mínimo (escalados por 100) de la CD "intra-bloque" (Ley constante).

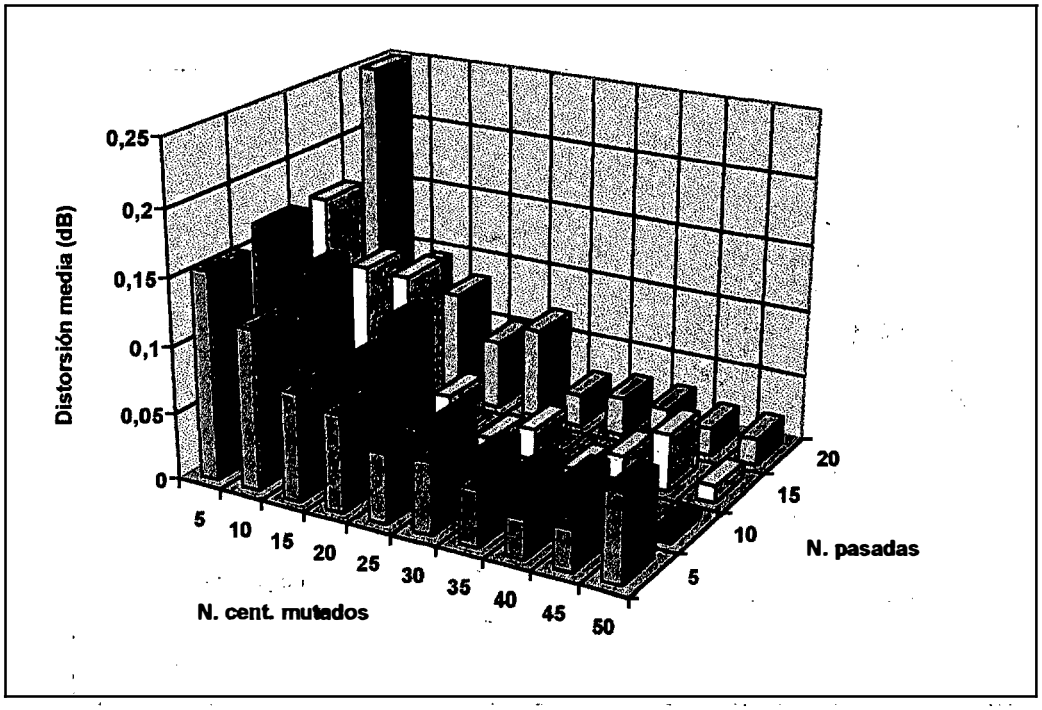

Fig. 5.14 Valores medios del "CD intrabloque"

Puede observarse que la ganancia del ARL optimizado frente al ARL de ley constante, disminuye conforme el número de centroides mutados aumenta. Por otra parte, el número de pasadas R más idóneo vuelve a resultar ser 15 para esta nueva Ley de Decrecimiento. De cualquier forma, en 39 de los 40 casos estudiados, la distorsión que proporciona el ARL optimizado es menor que la obtenida con el ARL lineal, hecho que constata la validez del mecanismo de refinado progresivo.

5.7- Análisis de resultados

Podemos resumir en los siguientes puntos todas las consideraciones extraídas de los resultados obtenidos en las simulaciones:

- los dos AGCV dan casi exactamente idéntica distorsión integral y final para todas las imágenes, BERs y tamaños del libro de códigos, lo que indica la robustez de este método con respecto al juego de parámetros y opciones que contiene;

- $AGCV_1$ supera en todos los casos a $AGCV_2$, aunque la diferencia es casi siempre ínfima;

- el funcionamiento de AIW se degrada de manera muy notoria con valores altos del BER. Esto confirma la idea de que los métodos de optimización en CV que no usan explícita o implícitamente la información estadística acerca del ruido de canal dada en el BER, no pueden competir con aquéllos que sí que lo hacen. En este punto es justo mencionar que cuando el BER no se conoce o varía de forma impredecible, puede ser ventajoso diseñar libros de código sin hacer referencia a esta variable. Sin embargo, esa no es la formulación general del problema que se acomete en esta Tesis;

- las prestaciones del TD han resultado excepcionalmente dependientes de la imagen considerada: para el caso de Lena, TD se ha comportado notoriamente peor que AGCV, ARL y GLA;

- la convergencia de estos tres últimos métodos es mucho más rápida que la del TD. Este hecho se deduce de los resultados del PSNR integral, consistentemente peores en TD que en los otros;

- los dos AGCV y ARL superan al GLA, tanto más cuanto mayor es el tamaño del libro de códigos;

- el mecanismo de refinado progresivo del ARL sujeto a la Ley de Decrecimiento lineal resulta un mecanismo acertado, en el sentido de que da lugar a menores distorsiones al compararlo con la versión del ARL en la que el número de centroides mutados permanece constante (ley de decrecimiento constante).

- AGCV y ARL presentan los mejores resultados, en general, para todos los casos considerados;

- con tamaños de libro de código pequeños (32 ó 64) estos dos métodos, AGCV y ARL, dan resultados prácticamente iguales. Sin embargo, con un tamaño de librería mayor (128 vectores) su comportamiento deja de ser uniforme y se hace dependiente de la magnitud del ruido en el canal. Así, cuando éste es pequeño (BER=1.0E-4) ARL supera al método genético en una o dos décimas de dB. Este resultado se invierte cuando el ruido de canal aumenta (BER=1.0E-2). Finalmente, sin embargo, cuando la PEB llega a 1.0E-1, ambos métodos obtienen de nuevo distorsiones semejantes.

- las distorsiones alcanzadas con AGCV, ARL y GLA para BER de 1.0E-1 son prácticamente idénticas;

Capítulo 6

DISCUSIÓN

Se pretende recoger en este capítulo algunas de las consideraciones surgidas del análisis de los datos obtenidos en la fase de ajuste previo y en los ensayos de simulación de los algoritmos AGCV y ARL.

Por otro lado, a lo largo del desarrollo de esta Tesis, fueron probadas varias alternativas que acabaron por ser abandonadas por no resultar competitivas. Pese a ello, merece la pena rescatar algunas de las ideas contenidas en ellos, intentando indagar, aun de forma superficial o incluso aventurada, en los motivos que les hicieron fracasar.

Además también los métodos GLA, TD, AIW suscitan varios comentarios, en particular, sobre los motivos de su elección y sobre los resultados que obtuvieron en las simulaciones.

Por último, el propio planteamiento del problema que se ataca en esta Tesis es objeto de una breve disquisición.

6.1- Sobre el método AGCV

6.1.1- Sobre las estrategias de evolución

Según lo expuesto en el capítulo 4, en las pruebas preliminares se contrastaron las estrategias de evolución Baldwiniana y Lamarckiana en el problema de CV bajo estudio. La supremacía de la segunda sobre la primera, reflejada en la Tabla 4.2 es palpable. El origen de este resultado es fácilmente interpretable a la luz de sus diferentes mecanismos (ver Figuras 2.46 y 2.47). Mientras que la estrategia Baldwiniana no incorpora en los individuos de la población las mejoras obtenidas en el proceso de aprendizaje (operador local GLA), la estrategia Lamarckiana sí que lo hace.

De esta forma, los individuos de la siguiente generación ya tendrán "ese camino andado". En el caso de la estrategia Baldwiniana, es la propia evolución genética la que tendrá que recorrerlo.

Sin embargo, el enfoque Baldwiniano sí que tiene razón de ser, aparte de una mayor cercanía a lo que ocurre en la Naturaleza. Por una parte, en cierto sentido no es del todo cierto que no permita a sus individuos incorporar genéticamente los logros de su aprendizaje. No lo hace directamente, pero sí de la manera indirecta que se explica a continuación.El operador local actúa aquí como un sensor remoto intentando inspeccionar el terreno en la dirección más favorable (hacia distorsiones localmente mínimas). El aprendizaje podrá ser incorporado a los individuos de la siguiente forma (ver Figura 6.1): el individuo i está situado en el punto p, mientras que el individuo j está en q, tratándose de un problema de minimización. Mediante el operador local, i se

desplaza hasta p', mientras que j, lo hace hasta q'. Aunque j se encuentra en una posición más ventajosa que i pues q<p, i será elegido con mayor probabilidad que j, ya que p'<q'. Esto no quiere decir que en la próxima generación, se encuentre en la posición p'.

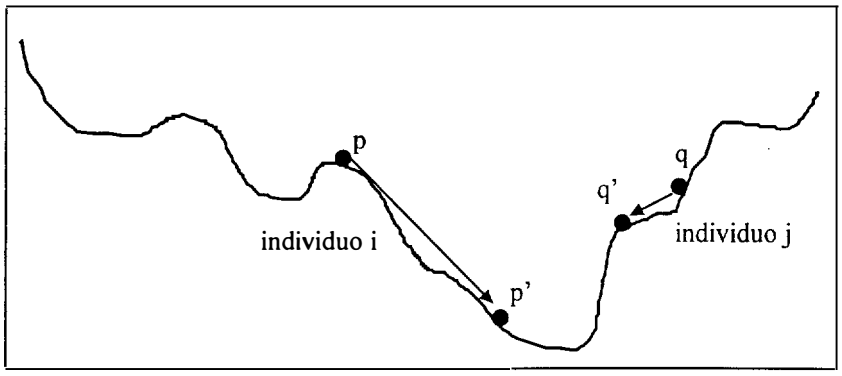

Fig. 6.1 Acción de un operador local en un proceso de búsqueda

Sin embargo, si debido a su alta adecuación se generan varios individuos en torno a p, puede que los procesos genéticos de mutación y cruce les conduzcan a la posición p' con cierta probabilidad.

Sin el mecanismo local, j sería más adecuado al entorno que i, por lo que el anterior aprendizaje no sería favorecido por la selección.(*)

Por otra parte, este modo de operación puede en algunos casos evitar una prematura convergencia de los individuos hacia mínimos locales poco convenientes, promovidos por el operador local.

Las pruebas realizadas confirmaron el mejor comportamiento de la estrategia Lamarckiana, pero no se podía haber asegurado "a priori" que esto fuera a resultar así.

Una ligera modificación de la estrategia Baldwiniana consiste en incorporar de alguna forma los costes del aprendizaje en la evaluación del individuo. Con ello se premiará a aquellos individuos que obtengan una adecuación grande a los mecanismos genéticos frente a aquéllos que lo logran por medio del aprendizaje. En último término se estará favoreciendo la "aceptación fenotípica de lo aprendido", en lo que se ha venido en llamar el "efecto Baldwin" [EC 96].

Este mecanismo también fue probado en nuestro problema de diseño en CV. La manera de llevarlo a cabo fue sumando a la función de coste (distorsión media de la cuantificación) un porcentaje de la disminución en la distorsión proporcionada por el optimizador GLA. De esta forma, aún no haciendo consideración alguna en la evaluación al coste del GLA, este mecanismo también alberga la esencia de premiar lo logrado genéticamente frente a lo aprendido.

Aunque las pruebas no fueron sistemáticas, los resultados no auguraban mayor éxito que el mecanismo Baldwiniano antes detallado, razón por la que fue desestimada esta opción.

(*) En todo esto hay que presuponer lo que Maley llama "correlación de vecindad", según la cual distancias cortas en el espacio fenotípico corresponden a distancias cortas en el espacio genotípico. [Maley 97].

6.1.2- Sobre la evaluación simplificada

Cuando el volumen de cómputo en la evaluación de los individuos de un genético es muy elevado, puede resultar interesante llevar a cabo dicha evaluación de forma aproximada, si esto conlleva una disminución significativa del mencionado coste. Más aún, esta evaluación simplificada puede realizarse en las primeras generaciones del algoritmo, haciendo que, a medida que éste vaya progresando, la aproximación vaya acercándose cada vez más al cálculo íntegro, para que los resultados finales no aparezcan sesgados por ella.

En [Grefenstette 85] se puede encontrar una explicación más detallada de esta técnica y un ejemplo de su aplicación al análisis de imágenes médicas.

En la optimización de CV la cantidad de cálculos requeridos es elevada. En particular, lo más costoso del proceso es el cálculo del vecino más próximo.

Según la RGVMP dada en (2.1.4.6.5), la subpartición correspondiente al vector $\bar{v}_s$ está dada por:

$$\mathrm{Cod}(\bar{v}_s) = \min_{i\in\{0,\ ,L-1\}} \left\{ \sum_{j=0}^{L-1} \mathit{Prob}(j / i) \left\| \bar{v}_s - \bar{c}_j \right\|_2 \right\} \qquad \textbf{(6.1.2.1)}$$

Esta operación ha de ser repetida sobre cada vector $\bar{v}_s$ del conjunto de entrenamiento (s= 0, N-1), en cada individuo, cada vez que se aplica esta ley, tanto para su evaluación, como para su optimización local mediante el GLA. Se desprende de aquí la conveniencia del ahorro de operaciones. De hecho, en el Apéndice 2, ya se explicaba una forma muy compacta para calcular la distorsión media del proceso. De este mismo desarrollo, también puede extraerse una forma para calcular la RGVMP menos costosa que la dada en (6.1.2.1), basadas en las siguientes expresiones:

$$\mathrm{Cod}(\bar{v}_s) = \min_{i\in\{0,\ ,L-1\}} \left\{ \sum_{d=1}^{M} \left(\beta_{id} - \alpha_{id} v_s(d)\right) \right\} \qquad \textbf{(6.1.2.2.a)}$$

donde

$$\alpha_{id} = 2\sum_{j=0}^{L-1} a_{ijd} \qquad \textbf{(6.1.2.2.b)}$$

$$\beta_{id} = \sum_{j=0}^{L-1} a_{ijd} c_j(d) \qquad \textbf{(6.1.2.2.c)}$$

$$a_{ijd} = \mathit{Prob}(j / i)\ c_j(d), \qquad \textbf{(6.1.2.2.d)}$$

y $v_s(d)$ y $c_j(d)$ son las componentes d-ésimas de los vectores $\bar{v}_s$ y $\bar{c}_j$, respectivamente.

Para reducir más aún los cálculos puede pensarse en intentar estimar el mínimo en las expresiones (6.1.2.1) ó (6.1.2.2) sin llegar a inspeccionar exhaustivamente sus L términos (particiones tentativas en las que situar al vector $\bar{v}_s$).

En este punto se puede introducir el concepto de metaparticiones, grandes regiones no solapadas en las que se divide el espacio de medida, y que albergan a varios vectores código (Figura 6.2).

Cada metapartición M_p tiene asociado un metacentroide $\hat{m}_p$, centroide a su vez de los vectores código incluidos en ella:

$$\hat{m}_p = \frac{1}{N_p} \sum_{q=0}^{N_p-1} \bar{c}_{pq}, \qquad p=0,\ H\text{-}1 \tag{6.1.2.3}$$

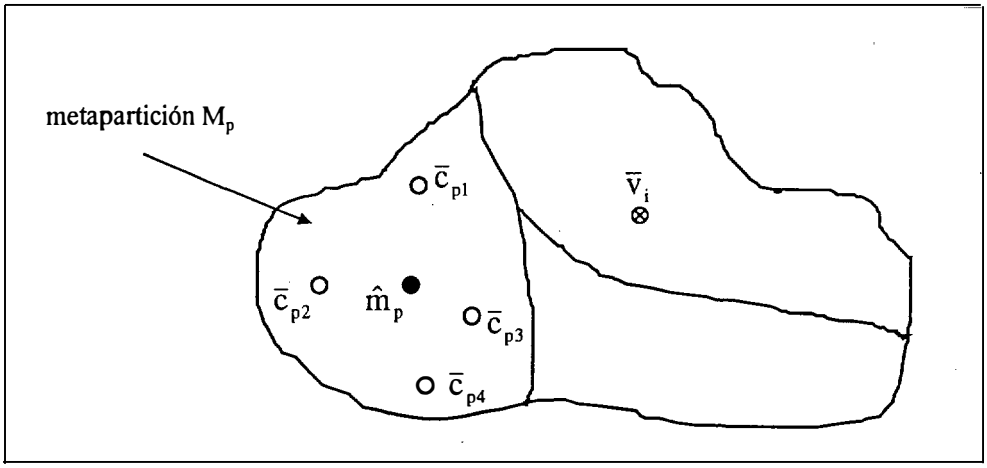

Fig. 6.2 Metaparticiones

donde H es el número total de metaparticiones y $\bar{c}_{pq}$ el vector q-ésimo M_p.

La idea fundamental de este cálculo aproximado consiste en que sólo se evaluarán los índices i de (6.1.2.1) ó de (6.1.2.2) correspondientes a regiones asociadas a vectores código incluidos en la metapartición más próxima a $\bar{v}_s$, donde la proximidad entre un vector y una metapartición se define como la distancia euclídea entre el vector y el metacentroide correspondiente a la metapartición.

Así en (6.1.2.1) ó en (6.1.2.2) sólo se evaluarán los índices i referidos a los centroides $\bar{c}_i$ pertenecientes a la metapartición M_r si resulta que ésta es la más próxima a $\bar{v}_s$ en el sentido de que:

$$\left\|\bar{v}_s - \hat{m}_r\right\|_2 \le \left\|\bar{v}_s - \hat{m}_p\right\|_2 \quad \forall p = 0,..H-1 \tag{6.1.2.4}$$

Otra posible simplificación de los cálculos, utilizada antes por otros autores [Farvardin 90] consiste en considerar que sólo va a existir, como mucho, un error en la transmisión de un código binario asociado a un vector código.

Según esto, las probabilidades de transición dadas en (2.1.4.3.4) serán ahora:

$$Prob(j / i) = \begin{cases} 1 - M\varepsilon, & \text{si } \|i,j\|_{ham} = 0 \iff i = j \\ \varepsilon, & \text{si } \|i,j\|_{ham} = 1 \\ 0, & \text{si } \|i,j\|_{ham} > 1 \end{cases} \tag{6.1.2.5}$$

donde M es el número de bits con que se codifican los vectores código ($M = \log_2(L)$).

En la expresión (6.1.2.1) esto supondría un importante ahorro, pues buena parte de los sumandos del sumatorio interno se anularían.

En la expresión (6.1.2.2), la mejora sería menos importante, ya que sólo afectaría a los coeficientes α_{id} y β_{id} a través de a_{ijd} y éstos son independientes del índice s que recorre los N vectores de entrenamiento.

Se hicieron pruebas con estas dos simplificaciones, incluyéndolas en la parte del cálculo de las particiones, pero no en el de la evaluación de las librerías de código, para evitar al máximo el sesgo de los resultados.

Sin embargo, ninguna de ellas logró mejorar los resultados en comparación con las configuraciones expuestas en temas anteriores. Aunque es difícil dar explicaciones a este respecto, quizás lo más razonable sea apelar a la complejidad del problema y a la intrincada y sensible relación entre la distorsión final y las variables involucradas.

Por último, conviene hacer notar que otros métodos de cálculo simplificado de vecinos más próximos, traídos de la Estadística y del Análisis Cluster, como el expuesto en [Nene 97], no son aplicables a nuestro problema. La razón es que la relación numérica entre un vector de entrenamiento ($\bar{v}_s$) y un vector código ($\bar{c}_i$), dada por:

$$\varphi(\bar{v}_s, \bar{c}_i) = \sum_{j=0}^{L-1} P_{ji} \left\| \bar{v}_s - \bar{c}_j \right\|_2 \qquad \textbf{(6.1.2.6)}$$

no cumple las premisas matemáticas de definición de distancia.

6.1.3- Sobre la representación de individuos

En el diseño de cuantificadores vectoriales mediante AG uno de las primeras cuestiones que aparecen es la de qué entidades van a tomarse como individuos de la población.

Cuando el paradigma es la optimización conjunta de vectores e índices, tal como se planteaba en 3.3.2, como es el caso de AGCV, la asignación de índices no entra en el proceso como una variable modificable y la optimización se realiza en el dominio de los vectores código o de las particiones.

La RGVMP impone unas particiones determinadas para una librería de vectores dada, por lo que si se considera la librería como variable independiente, la partición aparece como una variable dependiente de ésta. Alternativamente, si la RGC está presente, se intercambian estos papeles: ahora la variable independiente es la partición y la librería de códigos resulta dependiente de ésta.

Así pues, basta optimizar uno sólo de estos elementos y fijar el otro mediante la regla de optimalidad pertinente. Pero, ¿cuál de ellos elegir?

Las diferencias entre ambas opciones residen en el tipo de representación en el que se sustentan y en el modo en que los operadores genéticos actúan sobre los individuos.

Al igual que en [Choi 96], en nuestro AGCV se han elegido como individuos las librerías de código, que estarán representados por colecciones de L vectores de dimensión D (Figura 6.3):

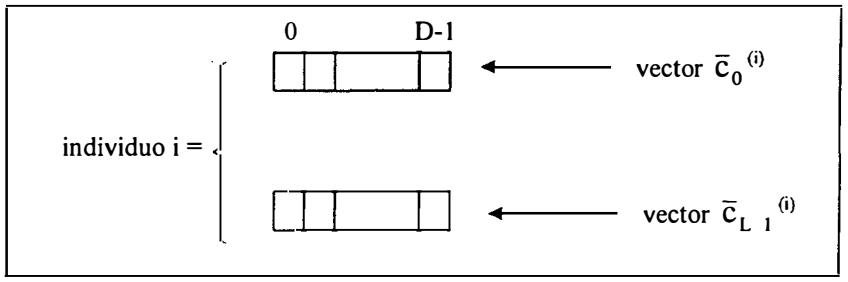

Fig. 6.3 Librerías de código como individuos de la población genética

Como ejemplos de mutación y cruce pueden tomarse los incorporados en AGCV y explicados en 4.2.1.2 y en 4.2.1.3. Ambos se aplican de forma directa sobre vectores código.

Si se hubieran elegido como individuos las particiones del espacio de vectores, éstos vendrían expresados como vectores de dimensión N, tomando cada componente, un valor entero entre 0 y N-1 (Figura 6.4):

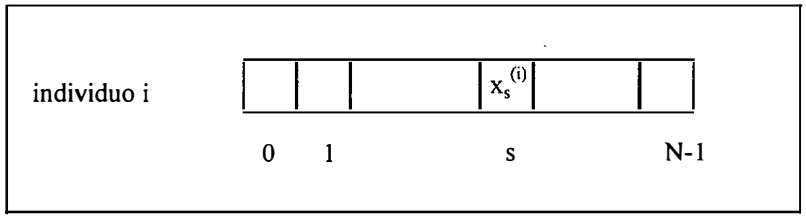

Fig. 6.4 Particiones como individuos de la población genética

Cada componente x_s expresaría la partición a la que se asociaría el vector de entrenamiento $\bar{v}_s$, con $0 \leq x_s \leq N-1$

Carece de sentido la codificación binaria de estos individuos, por cuanto las unidades mínimas de información son precisamente estas componentes x_s.

Un sencillo operador de cruce podría consistir en intercambiar uno a uno algunas de las componentes de los individuos cruzados (Figura 6.5).

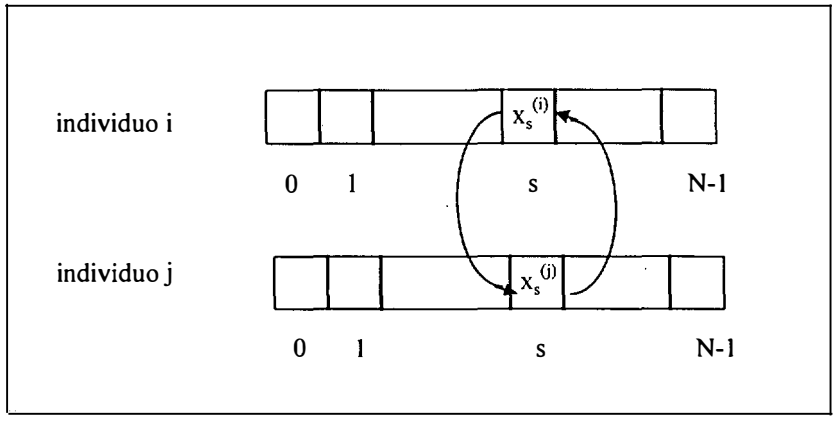

Fig. 6.5. Mecanismo de cruce cuando los individuos de la población genética son las particiones del espacio.

Este operador adolece de los mismos problemas de consistencia lógica que tenía el operador cruce aleatorio presentado en 4.2.1.2. Sin embargo, sería fácil la formulación de versiones del cruce constreñido o del cruce laxo adaptadas a este tipo de representación de los datos.

En cuanto al operador mutación, quizás lo más sencillo sería hacer variar al azar una o varias componentes del individuo, cambiándolas por enteros tomados aleatoriamente entre 0 y N-1, lo cual significaría, que el vector de entrenamiento en cuestión sería asociado a otra subpartición del espacio tomada de forma completamente aleatoria.

En esta Tesis se eligió la primera de las dos opciones de representación de individuos, por la razón de requerir estructuras de datos de menor tamaño. En ella, cada individuo es una matriz de números reales de dimensión LxD, mientras que en la segunda, los individuos son vectores de dimensión N.

En el conjunto de pruebas efectuados en esta Tesis, los valores que toman L, D y N son de: N=4096, L=32, 64 ó 128 y D=16.

Dado que el lenguaje en el que se han programado todos los algoritmos ha sido Matlab,(*) el cual maneja exclusivamente como tipo de datos matrices de números en punto flotante, la opción elegida resultaba ser, por tanto, 2, 4 u 8 veces más eficiente en este particular.

En lo concerniente a las posibilidades que una u otra representación pudieran brindar a los operadores genéticos, intuitivamente parece que los vectores de una librería son más fáciles de manipular a conveniencia, de forma coherente que las subparticiones del espacio. Sin embargo, la estrecha relación entre unas y otras, a través de las reglas de optimalidad, minimizan las diferencias que pudieran existir entre ambas.

La segunda de las opciones, en la que los individuos están constituidos por particiones tentativas, aunque fue implementada y probada para el caso no ruidoso, no llegó a ser objeto de pruebas de simulación sistemáticas en el caso ruidoso, por lo que es difícil aventurar si conduciría a resultados mejores o perores que la primera alternativa. Lo más probable no obstante, es que ambas resultaran muy semejantes, dada su ligazón a través de las reglas de optimalidad, antes comentada.

6.1.4- Sobre el cruce y la mutación

Poco más habría que añadir sobre estos dos operadores que no se haya dicho ya en el apartado 4.2.2, o que no quedara reflejado en la Tabla 4.4.

Es importante, no obstante, remarcar la sorprendente intrascendencia del cruce constreñido, hecho éste de difícil explicación, tanto más cuanto el operador mutación sí que muestra una marcada trascendencia en los resultados finales y un patrón de comportamiento bien definido. La mutación proporciona aleatoriedad al método, explorando nuevos puntos del espacio de búsqueda. Aunque de forma distinta, el cruce, sea alguno de los implementados en AGCV o de otro tipo, también introduce aleatoriedad, pero extrañamente, según los resultados vistos, ni beneficia ni perjudica globalmente al algoritmo.

(*) Aunque los programas han sido sometidos al proceso de compilación por medio de la herramienta Matlab Compiler, esto no afecta a la presente discusión.

Por otro lado, también se han probado algunos mecanismos que cambian dinámicamente las probabilidades de mutación y cruce. En concreto:

- una probabilidad de mutación que aumenta cada vez que el genético parece estancarse;
- probabilidades de cruce y mutación diferentes en cada individuo y dependientes de las prestaciones de sus padres, de acuerdo con el algoritmo dado en [Patnaik 96] y comentado en el apartado 2.2.8.2.

Nuevamente, aunque las pruebas no fueron exhaustivas, ninguna de estas estrategias condujo a ventajas dignas de consideración, por lo que también fueron descartadas.

6.2- Sobre la búsqueda en el algoritmo heurístico ARL

Habiéndose comentado los aspectos más importantes sobre este método en los capítulos 4 y 5, sería interesante reparar en un detalle, de no poca relevancia: el criterio de elección de los vectores código que van a ser reemplazados.

En el ARL, estos vectores código son elegidos atendiendo a su mayor distorsión parcial, definida en (4.3.1.2) como la distorsión correspondiente a la subpartición S_i:

$$D_i = \frac{1}{N} \sum_{k=0}^{N_i-1} \sum_{j=0}^{L-1} P_{ji} \left\| \bar{v}_{ik} - \bar{c}_j \right\|_2 \qquad \textbf{(6.2.1)}$$

Uno de los factores determinantes de la magnitud de esta distorsión, en particular cuando el BER es considerable(*), es el tamaño de la subpartición i, dado por N_i. Según esto, se estará penalizando a las subparticiones con mayor número de vectores código). Aunque por sí sólo y así formulado esto no parece claramente censurable, puede que sí lo sea si se contempla desde un ángulo inverso: favorecer la perpetuación de las substracciones que tienen un número pequeño de individuos quizás sea malgastar recursos.

Es posible que otras estrategias que limiten este sesgo, o lo eliminen del todo, normalizando por el tamaño de la subpartición N_i, redunden en mejores resultados.

Sin embargo, no se han hecho pruebas de simulación que aclaren este punto.

6.3- Sobre los algoritmos GLA, TD y AIW

Estos algoritmos, que han servido para la comparación de los métodos presentados en esta Tesis fueron seleccionados por los siguientes motivos:

- GLA es el algoritmo de diseño de CV clásico por antonomasia, lo cual, le convierte en el método más frecuentemente usado para contrastar técnicas nuevas.

- TD es un método más reciente, cuyo carácter global hacía presuponer que se trataría de un competidor más difícil de batir que el clásico GLA.

(*) El BER y la P_{ji} están relacionados mediante la expresión (2.1.4.3.4).

- AIW es un algoritmo que sólo optimiza las asignaciones de índices a partir de librerías de código previamente confeccionadas, generalmente optimizadas para condiciones de transmisión sin error. Esta estrategia es, en general, menos sofisticada que la optimización conjunta de vectores e índices, en las que se inscribe el resto de los métodos aquí comparados: AGCV, ARL, GLA y TD. Al incluir a AIW como tercer método de contraste se pretendía dar a las pruebas una perspectiva un poco más amplia, observando la influencia que la propia estrategia de búsqueda tiene en los resultados y cómo se ve afectada por el ruido.

El resultado más sorprendente de todos los obtenidos ha sido la mala adecuación de TD a los ensayos realizados. Las PSNR a que daba lugar se situaban muchas veces decenas de dB por debajo del resto. Además resultaba muy dependiente de la imagen de partida considerada y, por último, seguía un patrón de variación de la distorsión frente al BER que en muchos casos no era estrictamente creciente. Lo único que se puede argumentar al respecto es que los resultados publicados en [Miller 94] se realizaron sobre secuencias Gauss-Markov de primer orden y no en imágenes, siendo quizás el tipo y tamaño de los datos los principales causantes de su escaso éxito.

6.4- Sobre el planteamiento del problema

El problema planteado en esta Tesis ha sido el de encontrar libros de códigos y asignaciones que hagan mínima la función de distorsión (2.1.4.6.4), para unos datos de partida (imagen y BER) fijos y conocidos.

Obviamente, esta es una simplificación, posiblemente tosca, aunque muy manejable del que constituiría un planteamiento más realista del diseño de un cuantificador vectorial. Es más, la extrapolación de los resultados y de las conclusiones que se derivan de ellos al caso más general de imágenes no conocidas "a priori" y BER también desconocidos y quizás cambiantes, habría que hacerla con especial cuidado, sometiéndolas a pruebas que los avalaran.

A este respecto, en el siguiente capítulo se ofrece un análisis de la CV y de su optimización, cuando el BER no es conocido exactamente, sino sólo de forma probabilística.

En cualquier caso, el estudio en condiciones completamente controladas como las planteadas hasta ahora, permite comparar más fácilmente los métodos, extraer conclusiones puntuales acerca de su funcionamiento con mayor claridad y ahorrar de forma considerable los ya de por sí costosos cálculos.

Capítulo 7

ADENDA: CUANTIFICACIÓN VECTORIAL EN CANALES CUYA TASA DE ERROR POR BIT SÓLO SE CONOCE DE FORMA ESTADÍSTICA

7.1- Introducción

Como ya se ha comentado varias veces a lo largo de este trabajo, la distorsión en la CV en presencia de ruido de canal resulta de la combinación de dos factores:

- el propio proceso de cuantificación y
- el ruido añadido al decodificar bits recibidos de forma errónea.

Por otra parte, la probabilidad de error de un bit (Bit Error Rate o más compactamente y siguiendo la terminología empleada hasta ahora, BER) no tiene por qué ser fija, conocida o exactamente determinada, por lo que su caracterización será mucho más realista si se da en términos de una función densidad de probabilidad (f.d.p.(ε)) que con un valor concreto ε_0.

En este estudio teórico se quiere mostrar cómo llegar a la expresión analítica de la distorsión en CV de canal ruidoso, incluyendo de forma explícita la f.d.p.(ε). Seguidamente, los resultados hallados se particularizarán a tres sencillas situaciones de especial interés:

- ε constante fija y conocida;
- ε pequeña
- f.d.p.(ε) constante en un intervalo continuo cualquiera.

Finalmente se extenderá la RGC y la RGVMP a este caso más general, en el que ε no es fijo, sino que está dado por su f.d.p. La primera de estas reglas indicará cómo extraer el libro de códigos óptimo dada una partición del espacio de vectores. La segunda, marcará la partición óptima, a partir de un libro de códigos dado. Al igual que los casos de canal sin ruido (ε=0) y canal con BER constante (ε_0), la aplicación alternada e iterativa de ambas reglas constituirá un procedimiento localmente óptimo para el diseño de CV sujeto a las nuevas condiciones de contorno.

7.2- Planteamiento

El problema se plantea de manera análoga al caso de BER fijo, que se describía en 2.1.4.1, 2.1.4.2 y 2.1.4.3 y se esquematizaba en la Figura 2.19, que aquí repetimos (Figura 7.1). El vector $\bar{x}$ es codificado en el índice i, significando que el vector código

$\bar{c}_i$ es su vector representante (aquél con el que tendrá mínima distorsión). En el proceso de modulación en, el que está imbricado la función de asignación Π (ver Figura 2.20) se genera el símbolo b_i que será transmitido por el canal.

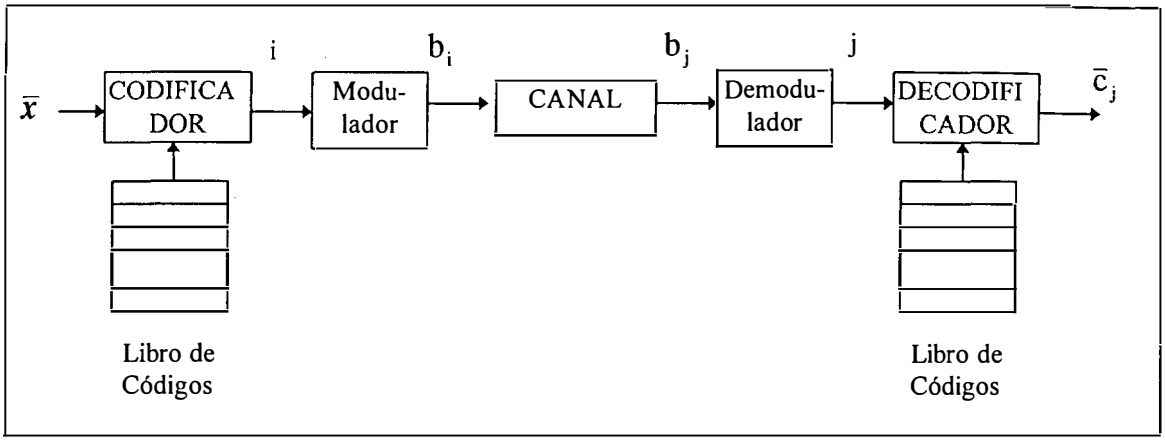

Fig. 7.1 Sistema de Cuantificación Vectorial en presencia de errores de canal.

En recepción, debido a los errores en la transmisión, en vez de b_i se recupera el símbolo b_j, que es transformado en el índice j que indica que el vector $\bar{c}_j$ será el que finalmente se extraiga a la salida del sistema.

El único aspecto diferente que debe considerarse en esta nueva situación es que la probabilidad P_{ji} de transición del símbolo b_i al b_j por efecto del ruido en el canal, ahora se calcularán a partir de f.d.p. del BER, en lugar de a partir de un BER fijo.

En efecto, si el BER es fijo (ε_0), y el canal simétrico, la probabilidad de que el índice i pase a ser el j viene dada por (2.1.4.3.4):

$$P_{ji} = \text{Prob}\left(b_j / b_i\right) = (1-\varepsilon)^{M-\|b_j,b_i\|_{ham}} \cdot \varepsilon^{\|b_j,b_i\|_{ham}} \qquad \textbf{(7.2.1)}$$

en la que M es el número de bits con los que se codifican los binarios i y j, ($L=2^M$).

Cuando el BER no es fijo, sino que está dado por la f.d.p.(ε), las probabilidades de transición se tendrán que computar de forma integral:

$$\hat{P}_{ji} = \int_0^1 (1-\varepsilon)^{M-\|b_j,b_i\|_{ham}} \cdot \varepsilon^{\|b_j,b_i\|_{ham}} \cdot \text{f.d.p.}(\varepsilon)\, d\varepsilon \qquad \textbf{(7.2.2)}$$

Obsérvese que el BER puede variar entre los valores 0 y 1, lo cual es coherente con su naturaleza de tasa de errores (número de bits erróneos, del total de bits enviados a través del canal).

7.3- Distorsión media del sistema

Si la la fuente de señal y el ruido son independientes, la distorsión media que resulta en este esquema es equivalente a la dada en (2.1.4.2.5), pero con las nuevas probabilidades de transición dadas en (7.2.2):

$$D = E\left\{\left\|\bar{x} - \bar{c}_j\right\|_2\right\} = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1} \hat{P}_{ji} \int_{S_i} f_X(\bar{x}) \cdot \left\|\bar{x} - \bar{c}_j\right\|_2 d\bar{x} \qquad \textbf{(7.3.3)}$$

Al igual que en (2.1.4.2.6), esta distorsión podrá separarse en dos términos uno ligado a la distorsión de fuente y otro a la de canal:

$$D = \sum_{i=0}^{L-1} \int_{S_i} f_X(\bar{x}) \left\|\bar{x} - \hat{c}_i\right\|_2 d\bar{x} + \sum_{i=0}^{L-1}\sum_{j=0}^{L-1} P_i \hat{P}_{ji} \left\|\bar{c}_{ij} - \hat{c}_i\right\|_2 \qquad \textbf{(7.3.4)}$$

siendo $\hat{c}_i$ el centroide (centro de masas) de la región S_i:

$$\hat{c}_i = \frac{\int_{S_i} \bar{x}\, f_X(\bar{x})\, d\bar{x}}{\int_{S_i} f_X(\bar{x})\, d\bar{x}} \qquad \textbf{(7.3.5)}$$

y P_i la probabilidad de que el vector $\bar{x}$ se encuentre en esta región:

$$P_i = \int_{S_i} f_X(\bar{x})\, d\bar{x} \qquad \textbf{(7.3.6)}$$

Por último, si se caracteriza la $f.d.p.(\bar{x})$ por el conjunto de vectores de entrenamiento, la distorsión media de (2.1.4.6.4) ahora tendrá la forma:

$$D = \frac{1}{N}\sum_{i=0}^{L-1}\sum_{k=0}^{N_i-1}\sum_{j=0}^{L-1} \hat{P}_{ji} \left\|\bar{v}_{ik} - \bar{c}_j\right\|_2 \qquad \textbf{(7.3.7)}$$

en la que $\bar{v}_{ik}$ es uno de los N_i vectores de entrenamiento codificados con el índice i.

Corolario: Las expresiones anteriores, en particular las (7.3.3), (7.3.4) y (7.3.7) son equivalentes, por completo, a las encontradas para el caso de BER fijo, con la sola excepción de que aparece el término $\hat{P}_{ji}$ en lugar de P_{ji}.

7.3.1- Casos particulares

Atendiendo al Corolario anterior, en los casos mostrados en este apartado, sólo prestaremos atención al término $\hat{P}_{ji}$, pues es el único que varía al cambiar las condiciones de ruido de canal y, por tanto, la f.d.p.(ε).

Caso 1: BER constante, fijo y conocido: $\varepsilon = \varepsilon_0$.

En estas circunstancias

$$f.d.p.(\varepsilon) = \delta(\varepsilon - \varepsilon_0) \qquad \textbf{(7.3.1.1)}$$

donde δ es la función delta de Dirac. Incluyendo esto en (7.2.2) se tiene:

$$\hat{P}_{ji} = \int_0^1 (1-\varepsilon)^{M-\|b_j,b_i\|_{ham}} \cdot \varepsilon^{\|b_j,b_i\|_{ham}} \cdot \delta(\varepsilon-\varepsilon_0)\, d\varepsilon$$

$$= (1-\varepsilon_0)^{M-\|b_j,b_i\|_{ham}} \cdot \varepsilon_0^{\|b_j,b_i\|_{ham}} \qquad \textbf{(7.3.1.2)}$$

resultado que ya sabíamos de (7.2.1).

Caso 2: BER pequeño y no constante.

Si las probabilidades de error son pequeñas, aún no siendo constantes o conocidas con precisión, puede asumirse que la probabilidad de que se encuentren dos o más errores en la transmisión de un símbolo sean nulas, por lo que sólo haría falta considerar las transiciones entre símbolos cuya distancia hamming sea de 0 ó de 1. En tal caso, se cumpliría la igualdad:

$$(1-\varepsilon)^{M-\|b_j,b_i\|_{ham}} \cdot \varepsilon^{\|b_j,b_i\|_{ham}} = \begin{cases} 1-M\varepsilon & si\ \|b_j,b_i\|_{ham}=0 \\ \varepsilon & si\ \|b_j,b_i\|_{ham}=1 \\ 0 & resto \end{cases} \qquad \textbf{(7.3.1.3)}$$

Según esto, $\hat{P}_{ji}$ podrá computarse como

$$\hat{P}_{ji} = \begin{cases} \int_0^1 (1-M\varepsilon)\cdot f.d.p.(\varepsilon)\, d\varepsilon & ,\ si\ \|b_j,b_i\|_{ham}=0 \\ \int_0^1 \varepsilon \cdot f.d.p.(\varepsilon)\, d\varepsilon & ,\ si\ \|b_j,b_i\|_{ham}=1 \\ 0 & ,\ resto \end{cases} \qquad \textbf{(7.3.1.4)}$$

de donde

$$\hat{P}_{ji} = \begin{cases} 1-M\eta_\varepsilon & si\ \|b_j,b_i\|_{ham}=0 \\ \eta_\varepsilon & si\ \|b_j,b_i\|_{ham}=1 \\ 0 & resto \end{cases} \qquad \textbf{(7.3.1.5)}$$

siendo η_ε es la esperanza de la variable aleatoria ε:

$$\eta_\varepsilon = E\{\varepsilon\} = \int_0^1 \varepsilon \cdot f.d.p.(\varepsilon)\, d\varepsilon \qquad \textbf{(7.3.1.6)}$$

Observamos cómo, para obtener $\hat{P}_{ji}$, en vez de calcular las integrales de (7.3.1.4) haciendo variar ε a lo largo de todo su intervalo de definición, simplemente se sustituye ε por su valor medio en el integrando (expresión 7.3.1.5).

Corolario: Esto equivale a todos los efectos a considerar que el BER es fijo y está dado por η_ε.

Caso 3: BER con densidad de probabilidad constante en un intervalo dado:

$$\text{f.d.p.}(\varepsilon) = \begin{cases} \dfrac{1}{\varepsilon_1 - \varepsilon_0} & , \; si \;\; \varepsilon_0 < \varepsilon < \varepsilon_1 \\ 0 & , \; resto \end{cases} \qquad \textbf{(7.3.1.7)}$$

La probabilidad de transición de (7.2.2) será ahora:

$$\hat{P}_{ji} = \int_{\varepsilon_0}^{\varepsilon_1} (1-\varepsilon)^{M-\|b_j,b_i\|_{ham}} \cdot \varepsilon^{\|b_j,b_i\|_{ham}} \cdot \frac{1}{\varepsilon_1 - \varepsilon_0} \, d\varepsilon \qquad \textbf{(7.3.1.8)}$$

Teniendo en cuenta que el desarrollo en serie de $(1-x)^p$ siendo p entero es:

$$(1-x)^p = \sum_{k=0}^{p} \binom{p}{k} \cdot (-x)^k \qquad \textbf{(7.3.1.9)}$$

la expresión (7.3.1.8) podrá desarrollarse de la siguiente manera:

$$\hat{P}_{ji} = \frac{1}{\varepsilon_1 - \varepsilon_0} \int_{\varepsilon_0}^{\varepsilon_1} \sum_{k=0}^{M-\|b_j,b_i\|_{ham}} \binom{M - \|b_j,b_i\|_{ham}}{k} \cdot (-1)^k \varepsilon^k \cdot \varepsilon^{\|b_j,b_i\|_{ham}} \cdot d\varepsilon$$

$$= \frac{1}{\varepsilon_1 - \varepsilon_0} \sum_{k=0}^{M-\|b_j,b_i\|_{ham}} \binom{M - \|b_j,b_i\|_{ham}}{k} \cdot (-1)^k \int_{\varepsilon_0}^{\varepsilon_1} \varepsilon^{k+\|b_j,b_i\|_{ham}} \cdot d\varepsilon$$

$$= \frac{1}{\varepsilon_1 - \varepsilon_0} \sum_{k=0}^{M-\|b_j,b_i\|_{ham}} \binom{M - \|b_j,b_i\|_{ham}}{k} \cdot (-1)^k \frac{\varepsilon_1^{\,k+\|b_j,b_i\|_{ham}+1} - \varepsilon_0^{\,k+\|b_j,b_i\|_{ham}+1}}{k + \|b_j,b_i\|_{ham} + 1}$$

(7.3.1.10)

7.3.2- Ejemplo

Para mostrar la validez de los anteriores desarrollos y el grado en que su consideración o no puede influir en el cálculo de la distorsión, se formula el siguiente ejemplo.

Más concretamente, y centrándonos en el Caso 3, en el que el BER se reparte estadísticamente de forma uniforme en un intervalo conocido, se trata de comprobar que, para un conjunto de entrenamiento, una librería de código y una asignación de índices dados, el cómputo de la distorsión a partir de (7.3.1.10) es correcto. Por otra parte, también se pretende observar que verdaderamente resulta una distorsión distinta de la que se hallaría considerando un BER constante, por ejemplo el valor medio del mencionado intervalo.

La imagen de partida es Lena, 256x256 pixeles con 8 bits/pixel de precisión, con la que se componen 4096 vectores de entrenamiento de dimensión 16, formados por bloques de 4x4 pixeles sin solapar. El tamaño del libro de códigos es de 32 vectores. Las condiciones del canal son las de un BER cuya f.d.p. es constante en el intervalo [0,0.1].

Se presupone la asignación "natural" de índices (ver apartado 2.1.4.1). El libro de códigos se obtiene ejecutando el algoritmo GLA tomándose un BER de 0.05.

Este libro de códigos constituye un dato de partida en este ejemplo y su mayor o menor adecuación a la imagen anterior no es relevante a efectos de las cuestiones que se abordan. Es por ello, que tampoco es trascendente el método utilizado para hallarla, ni siquiera el BER tomado.

Primeramente se calcula la distorsión teórica (D_t) según la expresión (7.3.1.10). Para comprobar la fiabilidad de esta expresión, se simula un canal de comunicaciones según la Fig. 7.2, sobre el que se ensaya el envío de la imagen anterior bajo las condiciones de ruido en el canal comentadas.

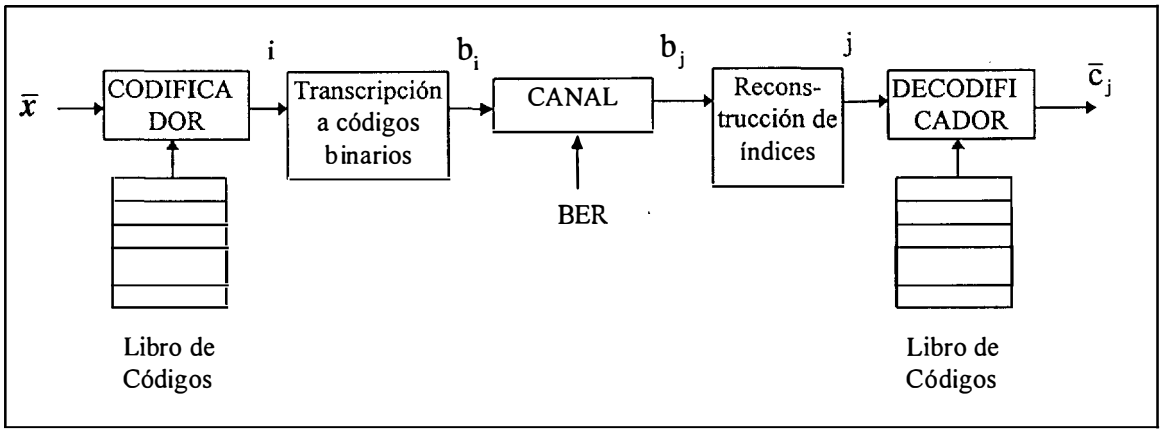

Fig. 7.2 Sistema de Cuantificación Vectorial en presencia de errores de canal.

Se lleva a cabo el siguiente protocolo de prueba:

Paso 1. Fijar los valores de ε_0, ε_1 y $\Delta\varepsilon$ (paso incremental del BER) en 0, 0.1 y 0.005, respectivamente. Fijar el valor del BER inicial: $\varepsilon = \varepsilon_0 + \frac{\Delta\varepsilon}{2}$

Paso 2. REPETIR hasta que BER no supere el valor final $\varepsilon = \varepsilon_1 - \frac{\Delta\varepsilon}{2}$

2.1- REPETIR PARA i=1 HASTA 100

2.1.1- Hacer pasar por el sistema anterior a cada vector $\bar{v}_s$ del conjunto de entrenamiento y computar a su salida la distorsión resultante según: $\|\bar{v}_s - \bar{c}_j\|_2$ s=0, .. N-1.

2.1.2- Obtener el promedio de estas distorsiones: D(ε,i)

2.2- Obtener el promedio de D(ε,i) a lo largo de las 100 repeticiones (Distorsión variable):

$$D_v(\varepsilon) = \frac{\sum_{i=1}^{100} D(\varepsilon, i)}{100}$$

2.3. Actualizar el BER según:

$$\varepsilon = \varepsilon + 0.01$$

Paso 3. Computar las distorsión simulada media (D_{sm}) como el promedio de los valores $D_v(\varepsilon)$.

Fig. 7.3 Protocolo para medir la distorsión simulada media.

Para cada valor de ε se llevan a cabo 100 simulaciones. Para lo que se pretende medir es indiferente el orden que tomen todas ellas, por lo que se hace que ε vaya tomando valores a partir de $\varepsilon_0 + \frac{\Delta\varepsilon}{2}$ hasta $\varepsilon_1 - \frac{\Delta\varepsilon}{2}$, con saltos de Δε. Con ello se consigue simular un ruido en el canal conforme lo descrito anteriormente, esto es, con un BER variable y aleatorio, uniformemente distribuido entre $\varepsilon_0 = 0$ y $\varepsilon_1 = 0.1$.

Este protocolo se repite 6 veces, obteniéndose sendos valores de distorsión simulada media.

Por otra parte, se utiliza la expresión (2.1.4.6.4), para observar qué distorsión resultaría con los datos anteriores y una probabilidad de error fija y de valor 0.05 (promedio de la $f.d.p.(\varepsilon)$ anteriormente considerada). Al valor así estimado se le llamará distorsión sesgada (D_s).

En la Tabla 7.1 se presenta la media muestral y desviación estándar de la distorsión simulada media, junto con la distorsión teórica y la sesgada.

Distorsión teórica:	6.1284 E-2	
Distorsión simulada media:		
-media muestral:	6.1323 E-2	
-desviación estándar:		2.8 E-5
Distorsión sesgada:	5.6949 E-2	

Tabla 7.1

Puede observarse en la tabla y en la figura que los resultados de las simulaciones coinciden plenamente con el obtenido de forma teórica a través de (7.3.1.10), y que ambos difieren sensiblemente respecto del hallado considerando un BER fijo y de valor 0.05 (media de la f.d.p. uniforme considerada para los casos anteriores).

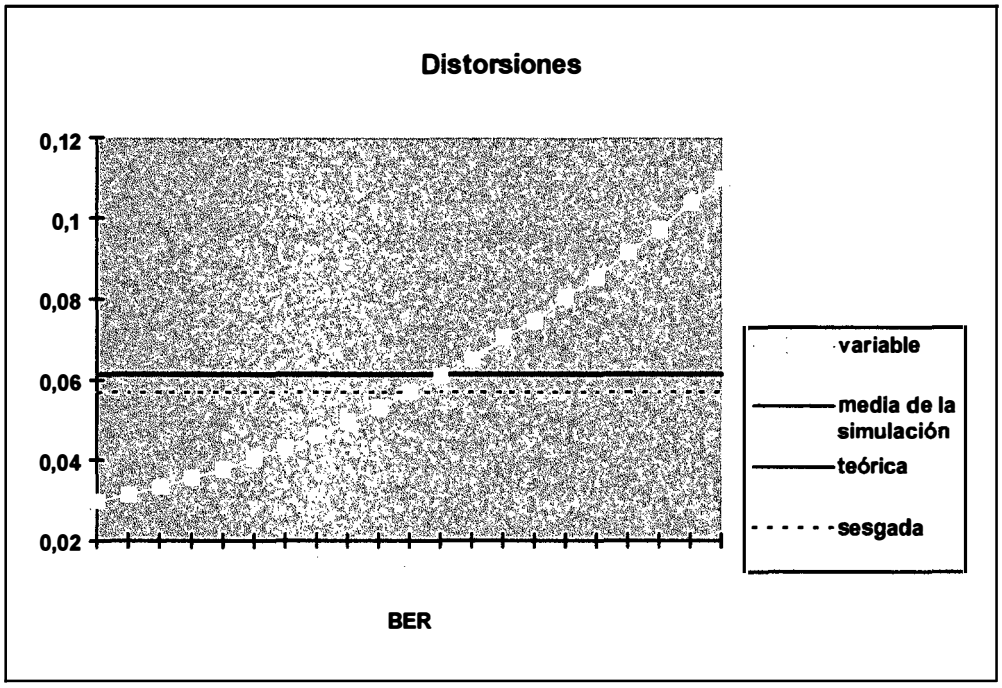

Fig. 7.4 Distorsiones variable ($D_v(\varepsilon)$), media de la simulación (D_{sm}), teórica (D_t) y sesgada (D_s).

7.4- Reglas de optimalidad

Se derivan a continuación sendas extensiones de las reglas RGC y RGVMP para la situación considerada en este capítulo en la que el BER es aleatorio y queda caracterizado estadísticamente por su f.d.p.

7.4.1- Regla Extendida de los Centroides (REC)

En el Apéndice 4 se obtiene una expresión para los vectores código óptimos en esta nueva situación:

$$\overline{c}_p = \frac{\sum_{i=0}^{L-1} \hat{P}_{pi} P_i \overline{c}_i}{\sum_{i=0}^{L-1} \hat{P}_{pi} P_i} \qquad \textbf{(7.4.1.1)}$$

donde, $\hat{P}_{ji}$ está tomada de (7.2.2):

$$\hat{P}_{ji} = \int_0^1 (1-\varepsilon)^{M-\|b_j,b_i\|_{ham}} \cdot \varepsilon^{\|b_j,b_i\|_{ham}} \cdot \text{f.d.p.}(\varepsilon)\, d\varepsilon \qquad \textbf{(7.4.1.2)}$$

donde, $\hat{P}_{ji}$ está tomada de (7.2.2):

$$\hat{P}_{ji} = \int_0^1 (1-\varepsilon)^{M-\|b_j,b_i\|_{ham}} \cdot \varepsilon^{\|b_j,b_i\|_{ham}} \cdot \text{f.d.p.}(\varepsilon)\, d\varepsilon \qquad \textbf{(7.4.1.2)}$$

En el caso de que la $\text{f.d.p.}(\bar{x})$ esté caracterizada por el conjunto de vectores de entrenamiento $V = \{\bar{v}_0, \bar{v}_1, \ , \bar{v}_{N\,1}\}$ la expresión (7.4.1.1) seguirá siendo válida, simplemente, teniendo en cuenta que ahora P_i y $\bar{c}_i$ estarán dados por:

$$P_i = \frac{N_i}{N} \qquad \textbf{(7.4.1.3)}$$

$$\bar{c}_i = \frac{1}{N_i} \sum_{k=0}^{N_i\ 1} \bar{v}_{ik} \qquad \textbf{(7.4.1.4)}$$

en la que $\bar{v}_{ik}$ es uno de los N_i vectores de entrenamiento codificados con el índice i.

7.4.2- Regla Extendida del Vecino Más Próximo (REVMP)

Extendiendo la expresión de esta regla que se da para el caso de BER fijo [Farvardin 90], a nuestro nuevo caso, se puede llegar a que las particiones óptimas del espacio de vectores para un libro de códigos y una función de permutación Π dados es:

$$S_i = \left\{ \bar{x} \mid \sum_{j=0}^{L-1} \hat{P}_{ji} \left\| \bar{x} - \bar{c}_j \right\|_2 \le \sum_{j=0}^{L-1} \hat{P}_{jp} \left\| \bar{x} - \bar{c}_j \right\|_2 , \ \forall\, p = 0,\ 1,..,\ L-1. \right\} \qquad \textbf{(7.4.2.1)}$$

En el caso de que la $\text{f.d.p.}(\bar{x})$ esté caracterizada por el conjunto de vectores de entrenamiento $V = \{\bar{v}_0, \bar{v}_1, \ , \bar{v}_{N\,1}\}$ esta nueva regla del vecino más próximo vendrá dada por:

$$V_i = \left\{ \bar{v} \mid \sum_{j=0}^{L-1} \hat{P}_{ji} \left\| \bar{v} - \bar{c}_j \right\|_2 \le \sum_{j=0}^{L-1} \hat{P}_{jp} \left\| \bar{v} - \bar{c}_j \right\|_2 , \ \forall\, p = 0,\ 1,..,\ L-1. \right\} \qquad \textbf{(7.4.2.2)}$$

Corolario: Las expresiones anteriores, en particular las (7.4.1.1), (7.4.2.1) y (7.4.2.2) son equivalentes por completo a las (2.1.4.4.2.1), (2.1.4.4.1.3) y (2.1.4.6.6) del caso con BER fijo, con la sola excepción de que ahora el término P_{ji} es sustituido por $\hat{P}_{ji}$.

7.5- Discusión

Varios comentarios surgen al respecto de lo expuesto en este capítulo:

- El algoritmo GLA se extenderá de manera natural al nuevo planteamiento, sustituyendo las reglas RGC y RGVMP por sus equivalentes extendidas: REC y REVMP. Esto simplemente equivale a utilizar $\hat{P}_{ji}$ en lugar de P_{ji}.

- Aunque las expresiones para hallar $\hat{P}_{ji}$ puedan ser más complejas que las de P_{ji} (ver expresión (7.3.1.10), por ejemplo), este juego de parámetros no depende de los vectores código, ni de las particiones inspeccionadas durante la búsqueda, por lo que es independientes del estado en que se encuentra el algoritmo y, consiguientemente, puede ser calculado al principio, sin tener que ser recalculado ninguna otra vez en el proceso. Es más, la integral (7.2.2), de no poder resolverse analíticamente, podrá estimarse de forma numérica, antes de comenzar la fase iterativa del algoritmo.

- Conviene remarcar, por si no hubiera quedado suficientemente claro, que en este capítulo no se formula ningún método nuevo para hallar libros de código óptimos, sino más bien se adecúa la función de distorsión media de la CV a la nueva situación planteada.

- En este sentido cabría comprobar si el GLA extendido con las nuevas REVMP y REC conduce a mejores resultados que el GLA normal. Aquí la función final de evaluación habrá de ser la expresada en (7.3.3) (o en (7.3.7) si el diseño se basa en datos empíricos). En este punto surge la cuestión de qué valores se da al BER fijo para ejecutar el GLA normal. En principio, el BER promedio, de acuerdo con la f.d.p. presente, parece la opción más sensata.

- También sería interesante estudiar el efecto de esta nueva función de coste sobre otros métodos de optimización de la CV, en particular, los métodos AGCV y ARL, aquí presentados. Nuevamente, los algoritmos implicados no tienen que ser alterados, solamente será preciso calcular inicialmente $\hat{P}_{ji}$ e incorporarla en lugar de P_{ji}.

Capítulo 8

CONCLUSIONES Y LÍNEAS ABIERTAS

8.1- Conclusiones

En esta Tesis se ha abordado el problema del diseño de cuantificadores vectoriales en presencia de ruido de canal, proponiéndose tres técnicas nuevas:

a- AGCV: un algoritmo genético, cuyos individuos son librerías de código tentativas y que incorpora un algoritmo GLA como optimizador local. El AGCV lleva a cabo una búsqueda explícita de la librería de códigos, mientras que la asignación de vectores a índices se optimiza de forma implícita.

b- ARL: un procedimiento heurístico que realiza la búsqueda de la librería de código óptima a través de repetidas ejecuciones del GLA sujetas a dos mecanismos:

- preservación de los mejores vectores código atendiendo a la menor distorsión asociada a ellos, y sustitución de los peores, por vectores de entrenamiento elegidos de forma aleatoria;
- disminución progresiva del número de vectores reemplazados a medida que avanza el algoritmo.

c- GHCV: algoritmo híbrido formado por un genético y un GLA actuando alternadamente. El genético optimiza la asignación de vectores a índices a partir de una librería de códigos entregada por el GLA, mientras que este último optimiza la librería después de que la asignación de índices haya sido modificada por el genético.

Los dos primeros han sido probados en la CV de tres imágenes diferentes, bajo distintas condiciones de ruido en el canal, que se ha supuesto binario, simétrico y sin memoria, y para distintos índices de compresión, determinados por el tamaño de los libros de código.

Para contrastar los resultados obtenidos se han usado tres reputados algoritmos: el GLA [Farvardin 90], el TD [Miller 94] y el AIW [Wu 93]. Todas las pruebas comparativas han sido efectuadas a igualdad de carga computacional.

El algoritmo GHCV no ha sido sometido a las pruebas anteriores y sólo se presenta como una técnica alternativa y complementaria a las dos primeras, aún sin explorar.

Las principales conclusiones a las que se ha llegado en este trabajo son:

- el algoritmo AGCV obtiene mejores resultados, tanto al final del proceso ($PSNR_f$) como en las etapas intermedias ($PSNR_i$), que las tres técnicas de contraste, en la práctica totalidad de los casos considerados. Además se presenta robusto frente al valor de sus parámetros internos y a la elección de las opciones que alberga.

- el algoritmo ARL también se muestra superior que los tres algoritmos de contraste. En realidad, las diferencias entre los resultados obtenidos por AGCV y ARL son nimias, excepto con librerías de tamaño grande (L=128), donde el comportamiento de ambas técnicas difiere y se hace en cierto sentido complementario, pues para bajos valores de ruido en el canal (BER=1.0E-4), ARL es superior, ocurriendo lo contrario cuando el ruido es mayor (BER=1.0E-2).

- La optimización secuencial de vectores e índices, representada en nuestros ensayos por el algoritmo AIW, ha mostrado su escasa eficacia en condiciones de ruido de canal apreciables (BER≥1.0E-3) en comparación con los métodos basados en la optimización conjunta de vectores e índices, en los que se engloban AGCV, ARL, GLA y TD.

En un plano diferente, se ha desarrollado una formulación analítica de la CV para el caso de que el canal esté caracterizado por un BER que no es fijo, sino que se da en términos probabilísticos mediante su f.d.p. A este respecto se han obtenido expresiones para la distorsión media del proceso y para las reglas de optimalidad con las que se pueden construir algoritmos de diseño. De este estudio se concluye:

- la utilidad de esta nueva formulación cuando los nuevos condicionamientos del problema,

- la escasa complejidad añadida en el diseño o evaluación de cuantificadores vectoriales, que su consideración supone.

8.2- Líneas abiertas

En el desarrollo de esta Tesis han surgido numerosas alternativas muchas de las cuales, por razones de tiempo y de concreción no han sido probadas suficientemente, quedando pendientes de un estudio más profundo y sistemático. Entre ellas:

- el propio algoritmo AHCV;

- un estudio más detallado del papel que juega la Ley de Decrecimiento en el algoritmo ARL. Mientras que el número de vectores descartados indica el balance entre la exploración y la explotación del espacio de búsqueda, la Ley de Decrecimiento marca la dinámica de este balance. El análisis de la influencia de este "juego exploración-explotación" sobre las prestaciones del algoritmo podría arrojar mucha y reveladora luz a la hora de elaborar leyes de decrecimiento más adecuadas o incluso óptimas;

- nuevos mecanismos para la evaluación simplificada de los individuos en el AGCV, al estilo de las propuestas explicadas en el apartado 6.1.2;

- utilización de las asignaciones como individuos del genético en el AGCV, tal como se proponía en 6.1.3;

- probabilidades de cruce y mutación adaptativas en el AGCV, según se indicaba en 6.1.4;

- otros operadores genéticos para el AGCV, como los explicados en los apartados 2.2.7.2 y 2.2.7.3;

- nuevas estrategias para la selección de los vectores código que van a ser reemplazados en el ARL, como las anunciadas en 6.2.

Por otra parte, el ensamblaje de varios de los métodos aquí expuestos o la combinación de algunos de sus principios, muy bien pudieran constituir el embrión de nuevos métodos de diseño de CV, en particular:

- en un esquema de optimización alternada de vectores e índices, utilizar el AGCV o el ARL como optimizador de la librería de códigos, y AIW u otro algoritmo de este estilo como el bloque de optimización de asignaciones;

- incorporar alguno o los dos mecanismos del ARL en los procesos de búsqueda del AGCV o en otros métodos de optimización global aplicados a CV.

Por último, las técnicas aquí desarrolladas o alguno de los conceptos desplegados en ellas pudieran utilizarse para abordar la solución de otro tipo de problemas cercanos al de CV aquí planteado. Por ejemplo en el diseño de:

- CV para señales desconocidas "a priori" y BER desconocidos y cambiantes (ver apartado 6.4);

- codificadores Trellis [Secker 92], [Nassar 93];

- CV constreñidos, predictivos, adaptativos o de tasa de bits variable (ver apartados 2.1.3.5, 2.1.3.6 y 2.1.3.7);

- Diseño conjunto del cuantificador vectorial y el modulador (MOR-VQ), [Skinnemoen 94], [Leung 97];

- codificadores CELP [Buzo 80], [Palival 93];

- clasificadores estadísticos [Miller 96];

- cuantificadores de color [Tasdizen 98].

Apéndice 1

ESTIMACIÓN DEL COSTE COMPUTACIONAL DE ALGORITMOS DE DISEÑO DE CV

En este Apéndice se obtendrán expresiones analíticas para estimar el coste computacional de varios algoritmos de optimización de CV, medido en términos de "número de iteraciones equivalentes del GLA" ("nieg").

A2.1- Primero de todo, se ha de determinar el coste computacional del GLA. Una iteración básica del GLA consiste en la simple aplicación de la regla RGVMP, seguida de la regla RGC. El número de multiplicaciones y sumas en punto flotante de la RGVMP es, respectivamente de:

$$P_{GNN} = L \cdot M(N + 2L) \quad \textbf{(A.1.1)}$$

$$S_{GNN} = L(M(N + 2L) + N) \quad \textbf{(A.1.2)}$$

donde L es el tamaño de la librería de códigos, N el número de vectores del conjunto de entrenamiento y M la dimensión de los vectores.

Las correspondientes a la RGC son:

$$P_{GC} = 2L^2M \quad \textbf{(A.1.3)}$$

$$S_{GC} = M(N + 2L^2) \quad \textbf{(A.1.4)}$$

Sumando (A.1.1) y (A.1.3), se obtiene una buena aproximación del número total de multiplicaciones realizadas en una iteración del GLA:

$$P_{GLA} = LM(N + 4L) \quad \textbf{(A.1.5)}$$

Sumando (A.1.2) y (A.1.4), se obtiene una buena aproximación del número total de sumas realizadas en una iteración del GLA:

$$S_{GLA} = NL + NM + MNL + 4L^2M \quad \textbf{(A.1.6)}$$

En todos los casos considerados en este trabajo, el tamaño de los vectores es de M=16, por lo que el primer término de la suma en (A.1.6) es 16 más pequeño que el tercer término. Además, dado que el libro de códigos es de tamaño L=32, 64, ó 128, el segundo término de la suma en (A.1.6) resulta tales veces más pequeño que el tercero.

De este modo y de forma aproximada, se pueden eliminar los dos primeros términos del sumatorio, teniéndose que el número de multiplicaciones en una iteración GLA es casi idéntico al de sumas. Su valor se aproximará mediante:

$$SP_{GLA} = LM(N + 4L) \quad \textbf{(A.1.7)}$$

También están presentes varias divisiones en punto flotante y otras operaciones matemáticas o de programación, pero no revisten un coste computacional comparable al de las sumas y productos.

A2.2- Se inspecciona a continuación el coste computacional de $AGCV_1$. La mayor parte de éste se encuentra en la regla RGVMP que es invocada una vez para cada individuo de la población dentro del proceso de evaluación y cada iteración del GLA en la optimización local. Llamando P_{OG1} y S_{OG1} al número total de multiplicaciones y sumas debidos a la regla anterior, aproximadamente se tiene que:

$$P_{OG1} = P\ I\ L\ M\ (N + 2L) \quad \textbf{(A.1.8)}$$
$$S_{OG1} = P\ I\ L\ (M\ (N + 2L) + N) \quad \textbf{(A.1.9)}$$

donde P es el tamaño de la población del genético e I es el número total de generaciones que tienen lugar en su ejecución. Según lo visto anteriormente, puesto que M=16, S_{OG1} puede aproximarse mediante:

$$S_{OG1} = P\ I\ L\ M\ (N + 2L) \quad \textbf{(A.1.10)}$$

Finalmente, el número de productos y sumas en punto flotante debido al proceso genético completo, se puede aproximar por:

$$SP_{OG1} = P\ I\ L\ M\ (N + 2L) \quad \textbf{(A.1.11)}$$

Por otro lado, se puede dar estimativamente el número total de productos y sumas debidas al proceso de optimización local:

$$SP_{OG2} = C \cdot SP_{GLA} = C\ L\ M\ (N + 4L) \quad \textbf{(A.1.12)}$$

donde C es el número total de iteraciones simples del proceso local GLA que tienen lugar a lo largo de toda la ejecución del algoritmo.

Finalmente, el coste computacional total del $AGCV_1$ medido en "nieg" resultará ser de:

$$\frac{SP_{OG1} + SP_{OG2}}{SP_{GLA}} = P\ I\ \frac{N + 2L}{N + 4L} + C \quad \textbf{(A.1.13)}$$

A.2.3- Las expresiones relativas al coste computacional del $AGCV_2$ son muy similares a las del $AGCV_1$. La única diferencia es el coste añadido por la operación de Cruce

Constreñido. El número total de productos y sumas relativos a este proceso puede aproximarse por:

$$P_{NG} = P\ I\ L^2 M \quad \textbf{(A.1.14)}$$

$$S_{NG} = 2\ P\ I\ L^2 M \quad \textbf{(A.1.15)}$$

En este caso y para facilitar el análisis, se asumirá que un producto en punto flotante es considerablemente más costoso que una adición, por lo que se podrá admitir que el número total de productos y adiciones en el $AGCV_2$ es de:

$$SP_{NG} = SP_{OG1} + SP_{OG2} + P\ I\ L^2 M \quad \textbf{(A.1.16)}$$

en cuyo caso, el coste total del proceso medido en "nieg" resultará de:

$$\frac{SP_{NG}}{SP_{GLA}} = P\ I\ \frac{N+2L}{N+4L} + C + P\ I\ \frac{L}{N+4L} = P\ I\ \frac{N+3L}{N+4L} + C \quad \textbf{(A.1.17)}$$

A.2.4- En cuanto al algoritmo ARL, dado que prácticamente su único proceder en cuanto a cálculo es la repetición del algoritmo GLA, si C representa el número total de estas repeticiones que tienen lugar a lo largo de toda su ejecución, el "nieg" total será de:

$$\frac{SP_{ARL}}{SP_{GLA}} = C \quad \textbf{(A.1.18)}$$

A.2.5- Para hallar una expresión del coste computacional del algoritmo TS [Miller94] se deben considerar dos partes fundamentales: una asociada a la Función de Asociación Débil (expresión (3.3.2.1.4)), y otra, a la regla RGC (expresión (3.3.2.1.10)). Una estimación del número de productos y sumas debidas a la primera puede ser:

$$P_{DA1} = I\ L\ (M\ (N+2L) + N) \approx I\ L\ M\ (N+2L) \quad \textbf{(A.1.19)}$$

$$S_{DA1} = I\ L\ (2L\ M + N(2+M) + L) \approx I\ L\ M\ (2L+N) \quad \textbf{(A.1.20)}$$

El número de productos y sumas debidas a la segunda está dado por:

$$P_{DA2} = I\ L\ N\ (L+M) \quad \textbf{(A.1.21)}$$

$$S_{DA2} = I\ L\ N\ (L+M) \quad \textbf{(A.1.22)}$$

Uniendo (A.1.19), (A.1.20), (A.1.21) y (A.1.22) se puede hallar el número total de multiplicaciones y sumas en punto flotante:

$$SP_{DA} = S_{DA1} + S_{DA2} = P_{DA1} = P_{DA2} = I\ L\ M\ (N+2L) + I\ L\ N\ (L+M)$$
$$= I\ L(2M\ N + (2M+N)\ L) \quad \textbf{(A.1.23)}$$

Ya que en todas las imágenes consideradas en esta Tesis, el número de vectores de entrenamiento es N=4096, entonces N>>2M, por lo que la última expresión puede aproximarse mediante:

$$SP_{DA} = I\ L\ N\ (2M + L) \qquad \textbf{(A.1.24)}$$

El número "nieg" total en el TD es, por tanto:

$$\frac{SP_{DA}}{SP_{GLA}} = \frac{I\ N\ (2M + L)}{M\ (N + 4L)} \qquad \textbf{(A.1.25)}$$

Apéndice 2

CÁLCULO EFICIENTE DE LA DISTORSIÓN GLOBAL EN LA CV

Cuando la RGVMP está presente, la distorsión media dada en (2.1.4.6.4) sólo depende de $\{C\}$ y de Π y puede expresarse como:

$$D(\{C\},\Pi) = \frac{1}{N}\sum_{s=0}^{N-1} \min_{i\in\{0,\ldots,L-1\}} \left\{ \sum_{j=0}^{L-1} Prob(j/i) \left\| \bar{v}_d - \bar{c}_j \right\|_2 \right\} \qquad \textbf{(A.2.1)}$$

donde s es una variable muda usada para indexar los vectores de entrenamiento y sumar la distorsión asociada a ellos.

El sumatorio interno de (A.2.1) puede reordenarse como sigue:

$$S_{si} = \sum_{j=0}^{L-1} Prob(j/i) \left\| \bar{v}_s - \bar{c}_j \right\|_2 = \sum_{j=0}^{L-1} Prob(j/i) \sum_{d=0}^{J-1} \left(v_s(d) - c_j(d) \right)^2 \qquad \textbf{(A.2.2)}$$

donde $v_s(d)$ y $c_j(d)$ son la componente d-ésima de los vectores $\bar{v}_s$ y $\bar{c}_j$, respectivamente. De manera sencilla la expresión anterior puede transformarse en:

$$\begin{aligned} S_{si} &= \sum_{j=0}^{L-1} Prob(j/i) \sum_{d=0}^{J-1} v_s^2(d) \quad 2c_j(d)\, v_s(d) + c_j^2(d) \\ &= \sum_{d=0}^{J-1} v_s^2(d) \sum_{j=0}^{L-1} Prob(j/i) - 2\sum_{d=0}^{J-1} v_s(d) \sum_{j=0}^{L-1} Prob(j/i)\, c_j(d) + \\ &\quad + \sum_{d=0}^{J-1} \sum_{j=0}^{L-1} Prob(j/i)\, c_j^2(d) \end{aligned} \qquad \textbf{(A.2.3)}$$

Notando que según la regla de probabilidad total,

$$\sum_{j=0}^{L-1} Prob(j/i) = 1 \qquad \textbf{(A.2.4)}$$

y llamando

$$a_{ijd} = Prob(j/i)\, c_j(d), \qquad \textbf{(A.2.5)}$$

se puede expresar S_{si} como:

$$S_{si} = \sum_{d=0}^{J-1}\left(v_s^2(d) - 2v_s(d)\sum_{j=0}^{L-1} a_{ijd} + \sum_{j=0}^{L-1} a_{ijd} c_j(d)\right) \qquad \textbf{(A.2.6)}$$

Incluyendo (A.2.6) en (A.2.1), finalmente se tiene que:

$$D(\{C\},\Pi) = \frac{1}{N}\sum_{s=0}^{N-1}\sum_{d=0}^{J-1} v_s^2(d) + \frac{1}{N}\sum_{s=0}^{N-1} \min_{i\in\{0,\ ,L\ \ 1\}} \left\{\sum_{d=0}^{J\ 1}(\beta_{id}\ \ \alpha_{id} v_s(d))\right\} \qquad \textbf{(A.2.7.a)}$$

donde

$$\alpha_{id} = 2\sum_{j=0}^{L\ 1} a_{ijd} \qquad \textbf{(A.2.7.b)}$$

$$\beta_{id} = \sum_{j=0}^{L-1} a_{ijd} c_j(d) \qquad \textbf{(A.2.7.c)}$$

Apéndice 3

ESPERANZA DE LA DISTORSIÓN MÍNIMA DE SUCESIVAS EJECUCIONES DEL ALGORITMO GLA

La distorsión final alcanzada por una ejecución del GLA presenta una varianza elevada debido a la aleatoriedad de la selección del libro de códigos inicial.

Considérese el siguiente experimento:

- se ejecuta el GLA P veces, tomando libros de códigos iniciales distintos y aleatorios cada vez;
- se extrae el mejor de los P libros de códigos finales;
- finalmente se obtiene la distorsión de este libro;

Todo esto equivale a seleccionar la menor de todas las distorsiones obtenidas en los P procesos.

Aunque este procedimiento presenta una varianza menor conforme P aumenta, todavía se pueden encontrar mejores métodos (con menor varianza) para estimar la distorsión mínima en P ejecuciones del GLA. El procedimiento que se deriva a continuación persigue este propósito.

Primero se ejecutan N ensayos independientes del GLA y se computa la distorsión final alcanzada en cada uno de ellos. A esta distorsión se le llama f(r), donde el índice r se refiere al ensayo (Figura A3.1). Además se registra el número de iteraciones de cada ejecución. El número de ensayos N debe ser lo suficientemente grande como para representar fielmente la estadística del proceso GLA.

Seguidamente se ordenan de menor a mayor las distorsiones obtenidas al final de los N ensayos, lo cual constituye una permutación de la variable r. Al nuevo índice se le llama n y a la función de ordenación Φ, por lo que

$$n=\Phi(r) \qquad \textbf{(A.3.1)}$$

La curva anterior redibujada con el nuevo eje de abscisas n se muestra en la Figura.A3.2.

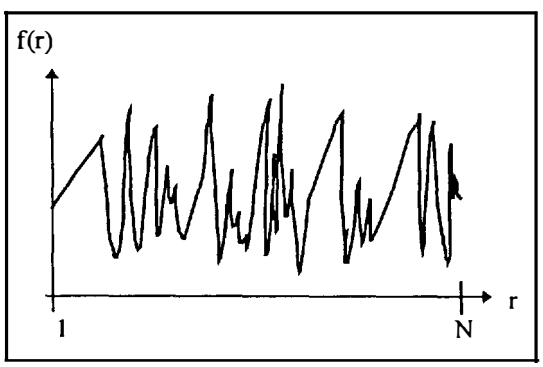

Fig.A3.1 Distorsión de N ensayos GLA

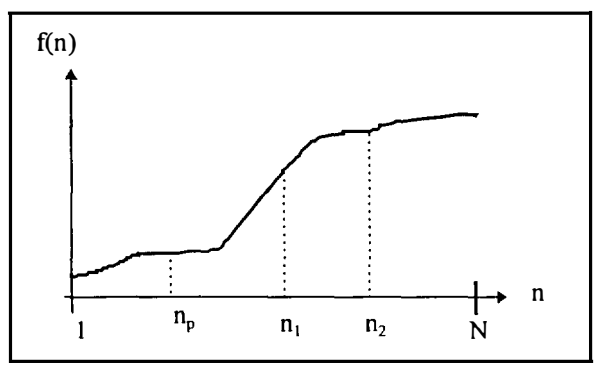

Fig.A3.2 Distorsión de N ensayos GLA ordenados

La acción de extraer de forma aleatoria e independiente P de los N ensayos GLA anteriores, tomando la mínima de sus respectivas distorsiones, equivale a generar aleatoriamente P números enteros uniformemente distribuidos en el intervalo [1,N], (llamémosles n_1, n_2, .. , n_p), elegir el menor de ellos y finalmente obtener su correspondiente distorsión ordenada. Para nuestros propósitos será conveniente extraer la esperanza de esta última distorsión, en función de N y P. Analíticamente,

$$EC\ (N,P) = \mathrm{E}\left\{f\left(min\left[n_1, n_2, \ , n_p\right]\right)\right\} \qquad \textbf{(A.3.2)}$$

Para dar con una expresión de esta esperanza, lo primero es hallar la función de probabilidades de n_1, n_2, n_p. Como todas ellas son independientes entre sí y su probabilidad es constante en su rango de definición y de valor 1/N, su función de probabilidad conjunta viene a ser:

$$\mathrm{Prob}[n_1, n_2, \ n_p] = \mathrm{Prob}[n_1]\ \mathrm{Prob}[n_2]\ .\ .\ \mathrm{Prob}[n_p] = \frac{1}{N}\cdot\frac{1}{N}\ \cdot\ \cdot\frac{1}{N} = \frac{1}{N^P} \qquad \textbf{(A.3.3)}$$

Según esto, el segundo miembro de (A.3.2) podrá expresarse como:

$$\mathrm{E}\left\{f\left(min\left[n_1, n_2, \ , n_p\right]\right)\right\} = \sum_{n_1=1}^{N}\sum_{n_2=1}^{N}\ \sum_{n_p=1}^{N} f\left(min\left[n_1, n_2, \ , n_p\right]\right) Prob\left[n_1, n_2, \ , n_p\right]$$

$$= \frac{1}{N^P}\sum_{n_1=1}^{N}\sum_{n_2=1}^{N}\ \sum_{n_p=1}^{N} \mathrm{f}\left(min\left[n_1, n_2, \ , n_p\right]\right) = \frac{1}{N^P}\ S \qquad \textbf{(A.3.4)}$$

siendo

$$S = \sum_{n_1=1}^{N}\sum_{n_2=1}^{N}\ \cdot\ \cdot\ \sum_{n_p=1}^{N} \mathrm{f}\left(min\left[n_1, n_2, \ , n_p\right]\right) \qquad \textbf{(A.3.5)}$$

El sumatorio múltiple de (A.3.4) está formado por N^p términos que se pueden visualizar en la matriz de la Figura A3.3, de la forma que se explica abajo.

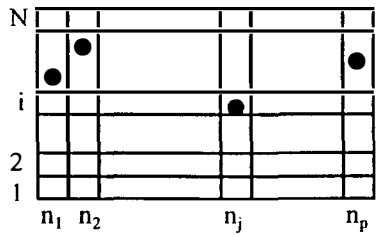

Fig.A3.3

Sólo se ha marcado una posición (valor entre 1 y N) en cada columna s, que se denotará como $v(n_s)$, o de forma más compacta, v_s. Cada conjunto diferente de N puntos corresponde a uno de los N^p términos del sumatorio anterior. Dado este conjunto, llamemos n_j a la variable cuyo punto marcado toma un valor mínimo, en el siguiente sentido:

$$\min[v_1, v_2, \ , v_p] = v_j \Leftrightarrow \begin{cases} v_j < v_s, & s = 1, 2, \ , j-1 \\ v_j \leq v_s, & s = j+1, \ , p \end{cases} \qquad \textbf{(A.3.6)}$$

Por su parte, llamemos i a este valor mínimo:

$$i = v_j = v(n_j) \qquad \textbf{(A.3.7)}$$

Ahora se podrá reorganizar S en un sumatorio, sólo a lo largo de las variables j e i:

$$S = \sum_{j=1}^{P} \sum_{i=1}^{N} f(i)\ R(i,j) \qquad \textbf{(A.3.8)}$$

donde R(i,j) es el número de términos en S (conjuntos permitidos en la Figura A3.3) que cumplen:

$$\begin{cases} \min[v_1, v_2, \ , v_p] = v_j \\ v_j = v(n_j) = i \end{cases} \qquad \textbf{(A.3.9)}$$

Para cada valor de i, los valores de v_j permitidos son los siguientes:

v_1= i+1, i+2, ,N (N-i valores permitidos)
v_2= i+1, i+2, ,N (N-i valores permitidos)
.
v_{j-1}= i+1, i+2, ,N (N-i valores permitidos)
v_j= i (1 valor permitido)
v_{j+1}= i, i+1, ,N (N-i+1 valores permitidos)
.
v_p= i, i+1, ,N (N-i+1 valores permitidos)

(A.3.10)

lo cual suma un total de $(N-i)^{j-1}(N-i+1)^{p-j}$ casos:

$$R(i,j) = (N-i)^{j-1}(N-i+1)^{p-j} \qquad \textbf{(A.3.11)}$$

Ahora, incluyendo (A.3.11) en (A.3.8), se tendrá,

$$S = \sum_{j=1}^{P} \sum_{i=1}^{N} f(i)(N-i)^{j-1}(N-i+1)^{p-j} \qquad \textbf{(A.3.12)}$$

A este resultado debe hacérsele una salvedad: cuando j=1 e i=N, el término $(N - i)^{j-1}$ está indeterminado. Sin embargo, en este caso particular,

$$min[n_1, n_2, \ldots, n_p] = N \qquad \textbf{(A.3.13)}$$

lo que significa que

$$n_1 = n_2 = \ldots = n_p = N \qquad \textbf{(A.3.14)}$$

Como en todo el resto de los casos en que i=N, y j≠1, los sumandos de S se anulan, la expresión (A.3.12) podrá expresarse según:

$$S = \sum_{j=1}^{P} \sum_{i=1}^{N-1} f(i)(N - i)^{j-1}(N - i + 1)^{p-j} + f(N) \qquad \textbf{(A.3.15)}$$

Finalmente,

$$EC(N,P) = \frac{1}{N^P}\left[\sum_{j=1}^{P} \sum_{i=1}^{N-1} f(i)(N - i)^{j-1}(N - i + 1)^{p-j} + f(N)\right] \qquad \textbf{(A.3.16)}$$

Apéndice 4

REGLA EXTENDIDA DE LOS CENTROIDES

A continuación se infiere la Regla Extendida de los Centroides, con la que se obtiene el conjunto óptimo de vectores código $C = \{\bar{c}_0, \bar{c}_1, , \bar{c}_{L-1}\}$ para una partición S_i y una función de asignación Π dadas, en el caso de que los canales sean ruidosos y se describan por una BER variable y aleatoria con f.d.p. conocida.

Bajo estas cincunstancias, la distorsión de cuantificación, está dada por la expresión (7.3.3), que repetimos a continuación:

$$D(C,\Pi) = E\left\{\left\|\bar{x} - \bar{c}_j\right\|_2\right\} = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1} \hat{P}_{ji} \int_{S_i} f_X(\bar{x}) \cdot \left\|\bar{x} - \bar{c}_j\right\|_2 d\bar{x} \qquad \textbf{(A.4.1)}$$

donde, a su vez, $\hat{P}_{ji}$ está tomada de (7.2.2):

$$\hat{P}_{ji} = \int_0^1 (1-\varepsilon)^{M-\|b_j,b_i\|_{ham}} \cdot \varepsilon^{\|b_j,b_i\|_{ham}} \cdot \text{f.d.p.}(\varepsilon)\, d\varepsilon \qquad \textbf{(A.4.2)}$$

Como se ve, $\hat{P}_{ji}$ no depende, ni explícita ni implícitamente de los vectores $\bar{c}_j$ elegidos. Así pues, los vectores código óptimos ($\bar{c}_p$) serán aquéllos que anulen la expresión:

$$\frac{\partial(D(C,\Pi))}{\partial \bar{c}_p} = \sum_{i=0}^{L-1} \hat{P}_{pi} \int_{S_i} f_X(\bar{x}) \cdot \frac{\partial\left(\left\|\bar{x} - \bar{c}_p\right\|_2\right)}{\partial \bar{c}_p} d\bar{x} \qquad (p=0, 1, , L-1) \qquad \textbf{(A.4.3)}$$

Por su parte, siguiendo consideraciones básicas del cálculo vectorial,

$$\frac{\partial\left(\left\|\bar{x} - \bar{c}_p\right\|_2\right)}{\partial \bar{c}_p} = -2\left(\bar{x} - \bar{c}_p\right) \qquad \textbf{(A.4.4)}$$

por lo que,

$$\int_{S_i} f_X(\bar{x}) \cdot \frac{\partial\left(\left\|\bar{x} - \bar{c}_p\right\|_2\right)}{\partial \bar{c}_p} d\bar{x} = -2 \int_{S_i} f_X(\bar{x}) \cdot \left(\bar{x} \quad \bar{c}_p\right) d\bar{x}$$

$$= \int_{S_i} f_X(\bar{x}) \cdot \bar{c}_p \, d\bar{x} - 2 \int_{S_i} f_X(\bar{x}) \cdot \bar{x} \, d\bar{x}$$

$$= 2 P_i \bar{c}_p - 2 \bar{c}_i P_i \qquad \textbf{(A.4.5)}$$

donde se ha tenido en cuenta que P_i es la probabilidad de que el vector $\bar{x}$ se encuentre en esta región, según aparecía en (7.3.6):

$$P_i = \int_{S_i} f_X(\bar{x}) \, d\bar{x} \qquad \textbf{(A.4.6)}$$

y que el centroide de la región S_i está dado por (2.1.3.2.2.1):

$$\bar{c}_i = \frac{\int_{S_i} f_x(\bar{x}) \bar{x} \, d\bar{x}}{\int_{S_i} f_x(\bar{x}) \, d\bar{x}} = \frac{\int_{S_i} f_x(\bar{x}) \bar{x} \, d\bar{x}}{P_i} \qquad \textbf{(A.4.7)}$$

Incluyendo el resultado de (A.4.5) en (A.4.3) e igualando a cero, se tiene que:

$$\sum_{i=0}^{L-1} \hat{P}_{pi} P_i \bar{c}_p = \sum_{i=0}^{L-1} \hat{P}_{pi} \bar{c}_i P \qquad \textbf{(A.4.8)}$$

Finalmente, los vectores código óptimos estarán dados por:

$$\bar{c}_p = \frac{\sum_{i=0}^{L-1} \hat{P}_{pi} P_i \bar{c}_i}{\sum_{i=0}^{L-1} \hat{P}_{pi} P_i} \qquad \textbf{(A.4.9)}$$

REFERENCIAS BIBLIOGRÁFICAS

[Anderberg 73] M. R. Anderberg, "Cluster Analysis for Applications". New York Academic, 1973.

[Anderson 70] N. Anderson, "Evolutionary Significance of Virus Infection". Nature, Vol. 227, 1970. pg. 1346-1347.

[Atal 93] B. S. Atal, V. Cuperman y A. Gersho (Editors). "Speech and Audio Coding for Wireless and Network Applications". Kluwer Academic Publishers. Boston, 1993.

[Back 91] T. Back, F. Hoffmeister y H. P. Schwefel, "A Survey of Evolution Strategies". En [ICGA 91], pg. 2-9.

[Baker 85] J. E. Baker, "Adaptive Selection Methods for Genetic Algorithms". En [ICGA 85], pp. 101-111.

[Bhaskaran 97] V. Bhaskaran y K. Konstantinides. "Image and Video Compression Standards. Algorithms and Architectures (2nd. Ed.)". Kluwer Academic Publishers. Boston, 1997.

[Black 93] J. V. Black, "Comparison of the Performance of Vector Quantizer Training Algorithms". Proceedings of the 3rd International Conference on Artificial Neural Networks. 1993. pg. 71-75.

[Brindle 81] A. Brindle, "Genetic Algorithms for Function Optimization" (Tesis Doctoral e Informe Técnico TR-81-2). Edmond University Alberta. 1981.

[Buckles 92] B. P. Buckles y F. E. Petry, "Genetic Algorithms". IEEE Computer Society Press. Los Alamitos, CA. 1992.

[Buzo 80] A. Buzo, A. H. Gray, R. M. Gray y J. D. Markel, "Speech Coding Based upon Vector Quantization". IEEE Trans. on Acoustic, Speech and Signal Processing. Vol 28, No. 5. Oct. 1980, pg. 562-574.

[Cetin 88] A. E Cetin y V. Weerackody. "Design Vector Quantizers using Simulated Annealing". IEEE Trans. on Circuits and Systems. Vol 31, No. 12. Dec. 1988, pg.1550.

[Choi 94] D. I. Choi y S. H. Park, "Self-Creating and Organizing Neural Networks".

[Choi 96] S. Choi y W. Keong Ng. "Vector Quantiser Design using Genetic Algorithms". Proceedings of the Data Compression Conference 1996, pp 430. IEEE Trans. on Neural Networks. Vol 5, No. 4. July. 1994, pg.561-575.

[Chong 96] E. K. P. Chong y S. H. Zak. "An Introduction to Optimization". John Wiley and Sons. New York, 1996.

[Cuperman 94] V. Cuperman, F.H. Liu y P. Ho, "Robust Vector Quantization for Noisy Channels Using Soft Decision and Sequential Decoding". European Transaction on Telecommunications and Related Technologies. Sept./Oct. 1994, pp.541-552.

[Darwin 76] C. Darwin, "El origen de las especies". Editorial Bruguera (5ª edición en Español) 1976. (La 1ª edición se publicó en lengua inglesa en 1859 con el Título "Origin of Species by Means of Organic Affinity").

[Davidor 89] Y. Davidor, "Analogous Crossover". En [ICGA 89], pg. 98-103.

[Davis 85] L. Davis, "Applying Adaptive Algorithms to Epistatic Domains". Proceedings of the International Joint Conference on Artificial Intelligence, 1985. pg. 162-164.

[Davis 89] L. Davis, "Adapting Operator Probabilities in Genetic Algorithms". En [ICGA 98], pg. 61-69.

[Davis 91] L. Davis, "Handbook of Genetic Algorithms". Van Nostrand Reinhold. New York. 1991.

[Dawkins 93] R. Dawkins "El gen egoísta. Las bases biológicas de nuestra conducta". Salvat (1ª edición en Español). 1993. (Traducción de la segunda edición en Inglés: "The selfish gene", 1989)

[De Jong 75] K. W. De Jong, "An Analysis of the Behaviour of a Class of Genetic Adaptive Systems". Tesis Doctoral, Universidad de Michigan, 1975.

[De Marca 87] J. R. B. De Marca y N. S. Noll, "An Algorithm for Assigning Binary Indices to the Codevector of a Multi-Dimensional Quantizer". Proceedings of the IEEE International Conference on Communications. Seattle WA, June 1987, pg. 1128-1132.

[**Díaz 96**] A. Díaz (Coordinador). "Optimización Heurística y Redes Neuronales". Ed. Paraninfo, 1996.

[EC 96] Evolutionary Computation. Vol. 4, No. 3, Fall 1996. Special Issue: "The Baldwin Effect".

[El Gamal 87] A. A. El Gamal, L. A. Hamachandra, Y. Shperling y V. Wei, "Using Simulated Annealing to Design good codes". IEEE Trans. on Information Theory. Vol 33, Jan. 1987, pg.116-123.

[Eshelman 91] L. J. Eshelman. "The CHC Adaptive Search Algorithm: How to have safe search when engaging in nontraditional Genetic recombination". En [Rawlins 91], pg. 265-283.

[Farvardin 87] N. Farvardin y V. Vaishampayan. "Optimal Quantizer Design for Noisy Channels: An Approach to Combined Source-Channel Coding". IEEE Trans. on Information Theory. Vol 33, No.6. Nov. 1987, pg. 827-838.

[Farvardin 90] N. Farvardin, "A Study of Vector Quantization for Noisy Channels". IEEE Trans. on Information Theory. Vol 36, No.4. July. 1990, pg. 799-809.

[Farvardin 91] N. Farvardin y V. Vaishampayan, "On the Performance and Complexity of Channel-Optimized Vector Quantizers". IEEE Trans. on Information Theory. Vol 37, No.1. January. 1991, pg. 155-160.

[Feo 89] T. A. Feo y M. G. C. Resende, "A probabilistic Heuristic for a computationally difficult set covering problem". Operations Research Letters, 8. 1989, pg. 67-71.

[Fogel 91] D. B. Fogel, "System Identification through Simulated Evolution: A Machine Learning Approach to Modeling". Ginn Press, Needham Heights MA, 1991.

[Forgey 65] E. Forgey, "Cluster analysis of multivariate data: efficiency versus interpretability of classification". Biometrics, 21. pg. 768, 1965 (Abstract).

[Fox 91] B. R. Fox y M. B. MacMahon, "Genetic Operators for Sequencing Problems". En [Rawlins 91], pg. 284-300.

[Furui 89] S. Furui. "Digital Speech Processing: Synthesis of Speech Signals". Marcel & Decker. New York, 1989.

[Gersho 92] A. Gersho y M. R. Gray. "Vector Quantization and Signal Compression". Kluwer Academic Publishers. Boston, 1992.

[Gibson 96] J. D. Gibson. "The Communication Handbook". CRC Press, 1996.

[Glover 86] F. Glover, "Future Paths for Integer Programming and Links to Artificial Intelligence". Computers and Operational Research No. 5, 533-549.

[Goldberg 85] D. E. Goldberg y R. Lingle, "Alleles, Loci and the Traveling Salesman Problem". En [ICGA 85], pg. 154-159.

[Goldberg 89] D. E. Goldberg. "Genetic Algorithms in Search, Optimization and Machine Learning". Addison-Wesley, Reading MA. 1989.

[Goldberg 91] D. E. Goldberg y K. Deb. "A comparative analysis of selection schemes used in Genetic Algorithms". En [Rawlins 91], pp. 78-81.

[González 95] A. I. González, M. Graña y A. D'Anjou, "An Analysis of the GLVQ Algorithm". IEEE Trans. on Neural Networks. Vol 6, No.4. July 1995, pg.1012-1016.

[Grefenstette 85] J. J. Grefenstette y J. M. Fitzpatrick, "Genetic Search with Approximate Function Evaluation". En [ICGA 85], pg. 112-120.

[Grefenstette 86] J. J. Grefenstette, "Optimization of Control Parameters for Genetic Algorithms". IEEE Trans. on Systems, Man and Cybernetics. Vol 16, No.1. 1986, pg. 122-128.

[Hansen 96] N. Hansen y A. Ostermeier. "Adapting Arbitrary Normal Mutation Distributions in Evolution Strategies: The Covariance Matrix Adaptation". En [ICEC 96], pg. 312-317.

[Haykin 94] S. Haykin. "Neural Networks: A Comprehensive Foundation". Macmillan, 1994.

[Hecht-Nielsen 91] R. Hecht-Nielsen, "Neurocomputing". Addison-Wesley. Reading, MA. 1991.

[Herdy 90] M. Herdy, "Application of the Evolution Strategy to Discrete Optimization Problems". Proceedings of the First International Conference on Parallel Problem Solving from Nature. Dortmund Alemania, 1990. pg. 188-192.

[Hertz 91] J. Hertz, A. Krogh y R. G. Palmer, "Introduction to the Theory of Neuron Computation". Addison-Wesley, New York. 1991

[Hoffmeister 91] F. Hoffmeister. "Scalable Parallelism by Evolutionary Algorithms". En "Applied Parallel and Distributed Optimization" Lecture Notes in Mathematical System and Economics (Editores M. Grauer y D. B. Pressmar). Springer-Verlag 1991.

[Holland 75] J. Holland. "Adaptation in Natural and Artificial Systems". Ann Arbor: University of Michigan Press, 1975.

[Hollstien 71] R. B. Hollstien, "Artificial Genetic Adaptation in Computation Control Systems". Doctoral Dissertation. University of Michigan. Dissertational Abstract International 32 (3), 1410 B (Universal microfilms No. 71-23, 773).

[ICEC 96] Proceedings of the IEEE International Conference on Evolutionary Computation. Nagoya, Japan. 1996.

[ICGA 85] Proceedings of the First International Conference on Genetic Algorithms. (Editor J. J. Grefensttete). L. Erlbaum Associates, Hillsdale, N.Y., 1985.

[ICGA 87] Proceedings of the Second International Conference on Genetic Algorithms. (Editor J. J. Grefensttete). L. Erlbaum Associates, Hillsdale, N.Y., 1987.

[ICGA 89] Proceedings of the Third International Conference on Genetic Algorithms. (Editor J. D. Schaffer). Morgan Kaufmann, San Diego, CA,.1989.

[ICGA 91] Proceedings of the Fourth International Conference on Genetic Algorithms. (Editores R. K. Belew y L. B. Booker). Morgan Kaufmann, San Diego, CA,.1991.

[ICGA 93] Proceedings of the Fifth International Conference on Genetic Algorithms. (Editor S. Forrest). Morgan Kaufmann, San Diego, CA,.1993.

[Ishibuchi 96] H. Ishibuchi y T. Murata, "Multi-Objective Genetic Local Search Algorithm". En [ICEC 96], pg. 119-124.

[Jaynes 89] E. T. Jaynes, "Information Theory and Statistical Mechanics". Papers of Probability, Statistics and Statistical Physics (De. R. S. Rosenkrantz) Dordretch, The Netherlands: Kluwer Academic Publishers (Reimpresión del artículo original de la revista Physical Review).

[Kangas 90] J. A. Kangas, T. K. Kohonen y J. T. Laaksonen, "Variants of Self-Organizing Maps". IEEE Trans. on Neural Networks. Vol 1, No.1. March. 1990, pg. 93-99.

[Karayiannis 95] N. B. Karayiannis y P. I. Pai, "Fuzzy Vector Quantization Algorithms and their Applications in Image Compression". IEEE Trans. on Image Processing. Vol 4, No.9. Sept. 1995, pp.1193-1201.

[Karayiannis 96] N. B. Karayiannis y P. I. Pai, "Fuzzy Algorithms for Learning Vector Quantization". IEEE Trans. on Neural Networks. Vol 7, No.5. Sept. 1996, pp.1196-1211.

[Karayiannis 97] N. B. Karayiannis, "A Methodology for Constructing Fuzzy Algorithms for Learning Vector Quantization". IEEE Trans. on Neural Networks. Vol 8, No3. May. 1997, pp.505-518.

[Katsavouridis 94] I. Katsavouridis, C.J. Kuo y Z. Zhang. "A New Initialization Technique for Generalized Lloyd Iteration". IEEE Signal Processing Letters, Vol. 1, No. 10, pg. 144-146. Oct. 1994.

[Kim 99] D. Kim y S. Ahn, "A MS-GS VQ Codebook Design for Wireless Image Communication using Genetic Algorithms". IEEE Trans. on Evolutionary Computation. Vol 3, No.1. April. 1999, pp.35-52.

[Kirkpatrick 83] S. Kirkpatrick, C. Gelatt y M. P. Vecchi. "Optimization by Simulated Annealing". Science, vol. 220, pg. 671-680, May 1983.

[Knagenhjelem 92] P. Knagenhjelem, "A Recursive Design Method for Robust Vector Quantization". Proceedings of the Signal Processing Applications and Technology Conference, 1992. pg. 948-954.

[Kohonen 89] T. K. Kohonen, "Self-Organization and Associative Memory". Springer-Verlag, Berlin 1989 (3ª Ed.).

[Kubota 96] N. Kubota, K. Shimojima y T. Fukuda, "The Role of Virus Infection in Virus-Evolutionary Genetic Algorithm". En [ICEC 91], pg. 182-187.

[Kumazawa 84] H. Kumazawa, M. Kasahara y T. Namekawa. "A construction of vector quantizers for noisy channels". Electronics and Engineering in Japan, vol 67-B. No. 4, pp. 39-47, 1984.

[Lee 97] D. Lee, S. Baek y K. Sung, "Modified K-means Algorithm for Vector Quantizer Design". IEEE Signal Processing Letters, Vol. 4, No. 1, pp. 2-4. Jan. 1997.

[Leung 97] C.S. Leung y L. W. Wan, "Transmission of Vector Quantized Data over a Noisy Channel". IEEE Trans. on Neural Networks. Vol. 8, No. 3, May 1997, pg. 582-589.

[Liepins 91] G. E. Liepins y W. D. Potter, "A Genetic Algoritm Approach to Multiple Fault Diagnosis". En [Davis 91], pg. 237-250.

[Linde 80] Y. Linde, A. Buzo y R.M. Gray. "An algorithm for Vector Quantization Design". IEEE Trans. on Communications. COM-28, Dec. 1980, pg. 84-95.

[Liu 87] Y. J. Liu. "Improving the Coding Design for Vector Quantization". Proceedings of the IEEE Military Communications Conference MILCOM-87, New York, 1987. pg. 556-558.

[Liu 93] F. H. Liu, P. Ho y V. Cuperman. "Joint Source and Channel Coding using a Non-Linear Receiver", Proceedings of the IEEE International Conference on Communications. Ginebra, 1993. pg. 1502-1507.

[Lloyd 57] S. P. Lloyd. "Least Squares Quantization in PCM" Notas no publicadas de Bell Laboratories. Presentado en parte en el Institute of Mathematical Statistical Meeting Atlantic City, Sept. 1957. Publicado más tarde en IEEE Trans. on Information Theory (Special Issue on Quantization). March 1982.

[Lukaszewicz 55] J. Lukaszewicz y H. Steinhaus. "On measuring by comparison". Zastos Mat. 1955. pg. 225-231.

[MacQueen 67] J. MacQueen, "Some methods for classifiction and analysis of multivariate observations". Proceedings of the 5th. Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pg. 281-296, 1967.

[Malanda 96] A. Malanda y A.R. Figueiras-Vidal, "Adaptive + Darwinian Approach for the Estimation and Tracking of Time Delays". Proceedings of EUSIPCO-96 (European Signal Processing Conference). Trieste, 1996. pg. 196-199.

[Malanda 98] A. Malanda y A. R. Figueiras-Vidal, "Hybrid Genetic Algorithm for Codebook Enhancement in Noisy Channel Vector Quantization". Proceedings of the IASTED Signal Processing and Communications Conference. Las Palmas de Gran Canaria, Feb. 1998. pp. 511-513.

[Maley 96] G. Mayley. "Landscapes, Learning Costs and Genetic Assimilations". En [EC 96], pp. 213-234.

[Mammen 90] C. P. Mammen y B. Ramamurthi. "Vector Quantization for Compression of Multichannel ECG". IEEE Trans. on Biomedical Engineering. Vol. 37, No. 9, Sept.. 1990, pg. 821-825.

[Martínez 93] T. M. Martínez, S. G. Berkovich y K. J. Schulten, "Neural-Gas Network for Vector Quantization and its Applications to Time-Series Prediction". IEEE Trans. on Neural Networks. Vol 4, No.4. July. 1993, pg. 558-569.

[Max 60] J. Max. "Quantizing for minimum distortion". IEEE Trans. on Info. Theory. March 1960, pg. 7-12.

[Michalewicz 92] Z. Michalewic, "Genetic Algorithms + Data Structures Evolution Programs". Springer Verlag. New York. 1992.

[Michavila 85] F. Michavila y L. Gavete, "Programación y Cálculo Numérico". Ed. Reverté, 1985.

[Miller 94] D. Miller y K. Rose, "Combined Source Channel Vector Quantization Using Deterministic Annealing". IEEE Trans. on Communications. Vol 42, No.2/3/4. Feb./March/April 1994, pg. 347-356.

[Miller 96] D. Miller, V. A. Rao, K. Rose y A. Gersho. "A Global Optimization Technique for Statistical Classifier Design". IEEE Trans. on Signal Processing. Vol 44, No.12. Dec. 1996, pg. 3108-3121.

[Muhlenbein 93] H. Muhlenbein, D. Schlierkamp-Voosen, "Optimal Interaction of mutation and crossover in the Breeder Genetic Algoritm". En [ICGA 93], pg. 648.

[Murata 96] T. Murata y H. Ishibuchi, "Positive and Negative Combination Effects of Crossover and Mutation Operators in Sequencing Problems". En [ICEC 96], pg. 171-175.

[Murthy 96] C. A. Murthy, S. Bandyopadhyay y S. K. Pal, "Genetic Algorithm-Based Pattern Classification: Relationship with Bayes Classifiers", En [Pal 96], pp. 127-144.

[Nassar 93] C. R. Nassar y R. M. Soleymani, "Codebook Design for Trellis Quantization Using Simulated Annealing". IEEE Trans. on Speech and Audio Processing. Vol 1, No.4. Oct. 1993, pg. 400-404.

[Nene 97] S. A. Nene y S. K. Nayar, "A Simple Algorithm for Nearest Neighbor Search in High Dimensions". IEEE Trans. on Pattern Anaysis and Machine Intelligence. Vol. 19, No.9. Sept. 1997, pg. 989-1003.

[Oliver 87] I. M. Oliver, D. J. Smith J. Holland, "A Study of Permutation Crossover Operations on the Traveling Salesman Problem". En [ICGA 87], pg. 224-230.

BIBLIOTECA

[Pal 93] N. R. Pal, J. C. Bezdek y E. C. K. Tsao, "Generalized Clustering Networks and Kohonen's Self-Organizing Scheme". IEEE Trans. on Neural Networks. Vol 4, No.4. July. 1993, pg. 549-557.

[Pal 96] S. K. Pal y P. P. Wang (Editores), "Genetic Algorithms for Pattern Recognition". CRC Press, Boca Ratón, 1996.

[Palival 93] K. K. Palival y B. S. Atal, "Efficient Vector Quantization of LPC Parameters at 24 Bits/Frame". IEEE Trans. on Sppech and Audio Processing. Vol 1, No.1. Jan. 1993, pg. 3-14.

[Pan 96] J.S. Pan, F. R. McInnes y M.A. Jack. "Application of Parallel Genetic Algorithms and Property of Multiple Global Optima to VQ Codevector Index Assignment for Noisy Channels". Electronics Letters on Line. 15 February 1996 (No. 19960194).

[Papantoni 79] P. Papantoni y R. M. Gray. "Robustness of Estimators on Stationary Observations". Ann. of Probability No. 7, Dec. 1979. pg. 989-1002.

[Patnaik 96] L. N. Patnaik y S. Mandavilli. "Adaptation in Genetic Algorithms En [Pal 96], pg. 45-64.

[Pitas 96] I Pitas, C. Kotropoulos, N. Nikolaidis, R. Yang y M. Gabbouj, "Order Statistics Learning Vector Quantizer". IEEE Trans. on Image Processing. Vol 5, No.6. June. 1996, pg. 1048-1053.

[Poggy 95] G. Poggy, "Applications of the Kohonen Algorithm in Vector Quantization". European Trans. on Telecomm. and Related.. Vol. 6, No. 2, 1995. pg. 191-202.

[Rao 96] K. R. Rao, J. J. Hwang. "Techniques and Standards for Image, Video and Audio Coding". Prentice-Hall, 1996.

[Rawlins 91] G. J. E. Rawlins (editor), "Foundations of Genetic Algorithms". Morgan Kaufmann, San Mateo, California, 1991.

[Richardson 89] J. T. Richardson, M. R. Palmer G. Liepins and M. Hilliard, "Some Guidelines for Genetic Algorithms with Penalty Functions". En [ICGA 89], pg. 191-195.

[Rose 92] K. Rose, E. Gurewitz y G. C. Fox. "Vector Quantization by Deterministic Annealing". ". IEEE Trans. on Information Theory. Vol 38, No.4. July. 1992, pg. 1249-1257.

[Schaffer 87] J. Schaffer y A. Morishima, "An Adaptive Crossover Distribution Mechanism for Genetic Algorithms". En [ICGA 87], pg. 36-40.

[Schaffer 89] J. Schaffer, R. Caruana, L. Eshelman y R. Das, "A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization". En [ICGA 89], pg. 51-60.

[Schaffer 91] J. Schaffer, L. J. Eshelman y D. Offutt, "Spurious Correlations and Premature Convergence in Genetic Algorithms". En [Rawlins 91], pg. 102-112.

[Schewefel 81] H. P. Schewefel, "Numerical Optimization for Computing Models". Chichester U. K. Wiley. 1981.

[Secker 92] P. Secker y A. Perkins, "Joint Source and Channel Trellis Coding for Line Spectrum Pair Parameters". Speech Communications 11, 1992. 149-158.

[Shannon 48] C.E. Shannon, "A Mathematical Theory of Communication". Bell Syst. Tech. Journal, No.27. pp. 379-423 y 623-656, 1948.

[Shannon 59] C.E. Shannon, "Coding theorems for a discrete source with a fidelity criterion". IRE National Convention Record, Part 4, pg. 142-163, 1959.

[Skinnemoen 94] H. Skinnemoen, "Combined Source-Channel Coding with Modulation Organized Vector Quantization (MOR-VQ)". Proceedings of the IEEE GLOBECOM. 1994, pg. 853-857.

[Smith 96] J. Smith y T. C. Fogarty. "Self Adaptation of Mutation Rates in Steady State Genetic Algorithm". En [ICEC 96], pg. 318-323.

[Syswerda 89] G. Syswerda, "Uniform Crossover in Genetic Algoritms". En [ICGA 96]. pg. 2-9.

[Tasdizen 98] T. Tasdizen, L. Akarun y C. Ersoy, "Color Quantization with Genetic Algorithms". Elsevier Science. Signal Processing: Image Communications No. 12, 1998, pg. 49-57.

[Wang 97] B. Wang y G. Yuan, "Compression of ECG Data by Vector Quantization". IEEE Engineering in Medicine and Biology, July/August 1997. pg. 23-26.

[Whitley 89] D. Whitley, T. Starkweather y D'A. Fuquay, "Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator". En [ICGA 89], pg. 133-140.

[Wu 93] H.S. Wu y J. Barba, "Index Allocation in Vector Quantization for Noisy Channels". Electronic Letters. Vol.29, No.15, 22nd. July 1993. pp. 1317-1319.

[Wu 94] H.S. Wu y L. Wuan, "Acceleration of the LBG Algorithm". IEEE Trans. on Communications. Vol 42, No.2/3/4. Feb/March/April. 1994, pg.1518-1523.

[Yair 92] E. Yair, K. Zegger y A. Gersho, "Competitive Learning and Soft Competition for Vector Quantizer Design". IEEE Trans. on Signal Processing. Vol 40, No.2. Feb. 1992, pg. 294-309.

[Zanakis 81] s. H. Zanakis y J. R. Evans, "Heuristic Optimization: How, When and How to use it". Interfaces, Vol. 11, No. 5, Oct. 1981.

[Zeger 87] K. Zeger, y A. Gersho, "Zero Redundancy Channel Coding in Vector Quantization". Electronic Letters. Vol.23, No.12, 4th June 1987. pg. 654-656.

[Zeger 90] K. Zeger, y A. Gersho, "Pseudo-Gray Coding". IEEE Trans. on Communications. Vol 38, No.12. Dec. 1992, pg. 2147-2158.

[Zeger 92] K. Zeger, J. Vaisey y A. Gersho, "Globally Optimal Vector Quantizer Design by Stochastic Relaxation". IEEE Trans. on Signal Processing. Vol 40, No.2. Feb. 1992, pg. 310-322.

[Zhang 93] B. S. Zhang, O. V. Patiakin, J. R. Leigh y G. Cain, "Comparison of Genetic Algorithms and Darwinian Adaptation for Search and Optimisation". Proceedings of the Natural Algorithms in Signal Processing Workshop, Essex U. K., Nov. 1993, pg. 3/1-3/8.

[Zheng 96] Y. Zheng y J. F. Greenleaf, The Effect of Concave and Convex Weight Adjustments on Self-Organizing Maps". IEEE Trans. on Neural Networks. Vol 7, No.1. Jan. 1996, pg. 87-96.