Decodificación de Códigos LDPC en canal de Rayleigh con Algoritmos Genéticos

Mónica C. Liberatori[1], Leonardo Arnone[2], Adriana Scandurra[3], Jorge Castiñeira Moreira[1]

[1] Laboratorio de Comunicaciones
[2]Laboratorio de Microcomponentes [3] Laboratorio de Bioingeniería
Facultad de Ingeniería de la Universidad Nacional de Mar del Plata,
Av. Juan B. Justo 4302, 7600 Mar del Plata, Argentina
mlibera@fi.mdp.edu.ar

Abstract. En este trabajo se propone la decodificación de Códigos Low Density Parity Check (LDPC) mediante un decodificador que utiliza una combinación de Algorimos Genéticos (GA, Genetic Algorithms) con Lógica Mayoritaria. La selección de GA obedece a la capacidad de los mismos para resolver problemas de optimización complejos, basándose en principios de la evolución natural de una población cuyos individuos se evalúan de acuerdo a una función de ajuste. El tipo de decodificación propuesto en este trabajo ha arrojado valores de Probabilidad Binaria de Error (BER, Bit Error Rate) comparables con la comportamiento del algoritmo tradicional de decodificación suma-producto para un canal de comunicaciones inalámbricas. La ventaja adicional de la decodificación propuesta frente a la tradicional es que no precisa conocer información de la relación Señal a Ruido en el canal.

Palabras clave: Decodificación, códigos LDPC, algoritmos genéticos, canal de Rayleigh.

1 Introducción

Los códigos de Matriz de Chequeo de Paridad de Baja Densidad (LDPC, Low Density Parity Check) propuestos por Gallager al principio de los ´60 [8, 11], han demostrado poseer un comportamiento cercano al límite de Shannon cuando utilizan un algoritmo de decodificación iterativo probabilístico [15,16].
Los códigos LDPC son códigos bloque lineales definidos por una matriz de paridad de baja densidad. El algoritmo de decodificación es fácilmente interpretable por medio de una representación gráfica conocida como grafo bipartito que también refleja la construcción del código. En el grafo, el proceso de decodificación se puede interpretar como el intercambio de información probabilística entre nodos símbolo y nodos de chequeo de paridad cuya relación queda determinada por la matriz de chequeo de paridad H. Cualquier vector perteneciente al código satisface todas las

condiciones de chequeo de paridad descriptas por dicha matriz. Cuando la longitud de las palabras código es lo suficientemente larga y la decodificación se realiza utilizando el método de decodificación probabilístico iterativo de Gallager, también conocido como algoritmo de suma-producto o de propagación de la certeza de la estimación (belief propagation), el comportamiento empírico de la tasa de error o BER es excelente [8, 13]. El algoritmo de decodificación es iterativo y dependiente del conocimiento del nivel de ruido presente en el canal [12].
En este trabajo se ha considerado la caracterización de un canal con respuesta al impulso variable en el tiempo, típica de canales inalámbricos debido a las cambiantes características físicas del medio. Se produce en estos canales un fenómeno conocido como desvanecimiento o fading que puede variar con el tiempo, la posición geográfica o la frecuencia de la señal y se puede modelar como un proceso aleatorio. En este sentido, el desvanecimiento tipo Rayleigh es un modelo estadístico apropiado para tener en cuenta el efecto de propagación multicamino en entornos inalámbricos de propagación troposférica y ionosférica, así como también en ambientes urbanos de gran densidad edilicia, donde no hay propagación dominante a lo largo de un camino directo (LOS, line of sight) entre transmisor y receptor [2].
Por su parte, los algoritmos genéticos, GAs, han sido utilizados en el campo de la electrónica y las comunicaciones, entre otros en el diseño de redes de datos [6], circuitos integrados de gran escala y códigos de distancia máxima [7]. En este trabajo se propone un esquema de decodificación de códigos LDPC basado en GAs y aplicado a la transmisión de datos sobre un canal tipo multicamino con desvanecimiento, resaltando que una de las principales ventajas es no precisar el conocimiento de la relación Señal a Ruido presente en el canal.

2 Consideraciones Generales

2.1 Códigos LDPC

Los códigos LDPC pertenecen a una clase muy poderosa de códigos tipo bloque lineales cuya matriz generadora G se utiliza para codificar un vector mensaje m de k elementos en una palabra código c de n elementos, tal que $c = G^T.m$. Cualquier palabra perteneciente al código satisface la condición de síndrome $H.c = 0$. El diseño del código se basa en la construcción de una matriz de paridad H especial. Se trata de una matriz con pocos elementos distintos de cero y muy desparramados entre sus filas y columnas que se suele referir como matriz *¨sparse¨*. Esta matriz satisface ciertas condiciones que proveen comportamiento óptimo en cuanto a la tasa de error, BER.
En los códigos LDPC binarios, tanto el vector mensaje m como el vector que representa la palabra código c, se definen sobre el campo binario, es decir sus componentes pertenecen al alfabeto discreto $\{0,1\}$. La transmisión de datos digitales es más conveniente cuando se realiza en formato polar, donde los bits se transmiten enviando señales del alfabeto discreto $\{-1,1\}$. Normalmente se asigna -1 al bit 0 y $+1$ al bit 1. De esta manera, el mensaje codificado c se convierte en un vector señal s. Luego de la transmisión y en presencia de ruido por reflexión multicamino, s se

transforma en el vector recibido $y_R = z.s + n$, cuyas componentes se pueden interpretar como tomadas de un conjunto de números reales $y_i \in \Re$ en el intervalo $[-\infty, \infty]$. En este trabajo se ha modelado el ruido como proveniente de un canal con desvanecimiento Rayleigh, por lo que el factor z se relaciona con este modelo adoptado.

El propósito de la decodificación es encontrar un vector d, considerado como una estimación del mensaje transmitido c, que satisfaga la condición $H.d = 0$

El algoritmo suma producto realiza una estimación de la Probabilidad A Posteriori (APP, A Posteriori Probability) de cada símbolo en función del símbolo recibido y las propiedades del canal [8, 11]. Es decir que se precisa conocer la relación Señal a Ruido presente en el canal.

2.2 Algoritmos Genéticos

Los Algoritmos Genéticos, GAs, son algoritmos de búsqueda propósito general que usan principios inspirados en la evolución natural genética [9]. Estos algoritmos se pueden aplicar a la resolución de problemas en los cuales una función objetivo podría ser discontinua, no diferenciable, estocástica o alineal. El GA mantiene una población de individuos que evolucionan de acuerdo a reglas de selección y operadores genéticos, tales como reproducción, cruce (crossover) y mutación. Se comienza con una población creada de manera aleatoria que se considera como conjunto de posibles soluciones del problema. Luego repetidamente se modifica esta población de tal manera que evoluciona hacia una solución óptima [10, 14]. Se asigna una medida a cada individuo de la población de acuerdo al valor que toma en cuanto a la función de ajuste, conocida como función de fitness. En la reproducción se intenta fijar la atención en los individuos de mejor valor de ajuste. A su vez, los operadores de cruce y mutación perturban esos individuos proveyendo heurística general para nuevas exploraciones. La operación de cruce se realiza para que los nuevos individuos posean las partes buenas de los viejos, suponiendo que serán mejores en términos de la función de ajuste. El propósito de la operación de mutación es evitar caer en mínimos locales.

Aunque simples desde un punto de vista biológico, se trata de algoritmos lo suficientemente complejos como para brindar mecanismos de búsqueda adaptables y robustos. De este modo, ante un problema particular, se debe realizar una evaluación ad-hoc de la función de ajuste, siendo la comportamiento del GA función de la adaptabilidad de los valores de los parámetros de la misma al caso particular bajo prueba [8, 17, 18].

Se destaca que en el caso del decodificador propuesto no se precisa conocer la relación Señal a Ruido presente en el canal

2.3 Canal de Rayleigh

El desvanecimiento tipo Rayleigh se debe a la recepción multicamino. La antena móvil recibe un gran número, N, de ondas reflejadas por diversos obstáculos y, debido a efectos de cancelación, la potencia instantánea recibida se entiende como

una variable aleatoria dependiente se la ubicación. En este tipo de entornos es razonable pensar que al transmitir un impulso, el mismo será recibido como un tren de impulsos.
Entonces, en términos generales la señal transmitida se puede representar como:

$$s(t) = \Re[s_l(t)e^{j2\pi f_c t}] \quad (1)$$

Dado que existen múltiples caminos de propagación, asociado con cada uno se considerará un factor de atenuación, $\alpha_n(t)$, y un retardo de propagación, $\tau_n(t)$, que resultan variantes en el tiempo como resultado de los cambios en la ¨estructura¨ del medio. De este modo, la señal pasa-banda recibida puede expresarse como:

$$y(t) = \sum_n \alpha_n(t)s(t - \tau_n(t)) \quad (2)$$

El retardo de propagación también se puede asociar a un cambio de fase de la señal $\theta_n(t) = 2\pi f_c \tau_n(t)$. Es decir que la fase de cada camino puede cambiar en 2π radianes cuando el retardo varía en $1/f_c$. Como la frecuencia está asociada a una longitud de onda que resulta mucho más pequeña que la distancia entre dispositivos, es razonable suponer que la fase es una variable aleatoria uniformemente distribuida entre 0 y 2π y que las fases de cada camino son independientes entre sí:

$$p(\theta) = \frac{1}{2\pi} \quad -\pi \le \theta \le \pi \quad (3)$$

Por otra parte, cuando la cantidad de caminos múltiples es grande, podemos aplicar el Teorema del Límite Central, donde cada camino puede modelarse como una variable aleatoria gaussiana compleja circularmente simétrica, con el tiempo como variable. Este modelo se conoce como modelo de canal de desvanecimiento de Rayleigh.
La variable aleatoria circularmente simétrica compleja gaussiana tiene como componentes real e imaginaria a dos variables aleatorias gaussianas idénticamente distribuidas (iid) e independientes de valor medio 0. La magnitud de dicha variable posee una distribución del tipo Rayleigh, razonable para entornos con múltiples reflectores:

$$p(z) = \frac{z}{\sigma^2} e^{-\frac{z^2}{2\sigma^2}} \quad , z \ge 0 \quad (4)$$

La señal recibida en este canal es de la forma:

$$y_R = z.s + n \quad (5)$$

donde s es la palabra codificada con símbolos en el alfabeto $\{-1,1\}$, z es un factor de escala complejo correspondiente al canal multi-camino mencionado previamente y n es una variable aleatoria que representa ruido blanco gaussiano aditivo (AWGN) con función densidad de probabilidad:

$$p(n) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(n-\mu)^2}{2\sigma^2}} \quad \mu = 0, \sigma^2 = \eta/2 \tag{6}$$

siendo η la densidad espectral de ruido. Si se supone que el canal z es conocido, en el receptor se realiza una operación de ecualización dividiendo cada símbolo recibido por el valor de z :

$$y = y_R / z = (z.s + n)/z = s + \tilde{n} \tag{7}$$

donde $\tilde{n}$ es el ruido aditivo escalado por el coeficiente del canal.

2 Decodificador LDPC con GA con Lógica Mayoritaria

Se genera una muestra de 1000 palabras codificadas contaminadas con ruido. El decodificador propuesto realiza tres pasos para tomar una decisión de decodificación: primero calcula el vector síndrome asociado al vector recibido, luego aplica el algoritmo genético obteniendo m candidatos por cada palabra, según la cantidad de veces que se elige correr el GA. Por último, se realiza la decisión final por lógica mayoritaria entre los candidatos del paso previo [5]. Un decodificador similar ha sido propuesto para el caso de un canal con ruido blanco gaussiano, aunque la función de ajuste ha sido modificada y los valores de los parámetros de corrida del GA se han ajustado para obtener resultados comparables a los de la decodificación tradicional [1, 4].

2.1 Cálculo del vector síndrome

Primero se digitaliza la palabra recibida sencillamente por decisión *hard*, es decir por comparación de los valores $y_i \in \Re$ con un umbral nulo. Este vector recibido modificado se denomina y_{hard} y se utiliza para calcular vector síndrome asociado a la palabra recibida. Si el cálculo de síndrome es el vector nulo, entonces se satisface la condición de que la palabra recibida pertenece al código y se frena el proceso de decodificación, considerando el vector y_{hard} como la palabra codificada enviada y procediendo a tomar otra palabra recibida de la muestra. Si el cálculo del síndrome genera un vector distinto de cero se procede a aplicar el algoritmo genético para obtener un vector decodificado.

2.2 Aplicación del GA

Cuando el vector síndrome calculado da una palabra distinta del vector nulo significa que la palabra recibida contiene errores y debe ser corregida. Entonces se procede a la aplicación del GA. El algoritmo genera una población de P posibles candidatos o individuos p. Cada individuo es un vector p con componentes p_i tomados de una distribución aleatoria uniforme en el intervalo de números reales $[0,1]$. Al aplicar el GA se obtiene el mejor candidato q entre todos los vectores del conjunto P. La selección se produce por búsqueda del mínimo de una función de ajuste adaptada al problema.

En la función de ajuste elegida en este trabajo intervienen dos factores: la distancia euclidiana entre el vector recibido y el candidato ofrecido por el algoritmo y el valor del vector de síndrome asociado al candidato.

En cualquier esquema de decodificación tradicional la distancia entre una palabra recibida y alguna otra seleccionada por el proceso de decodificación es una medida asociada a la justicia de la elección, siendo más cercana a lo verdadero cuanto menor sea este valor, es decir más próximos se encuentren ambos vectores entre sí.

En cuanto al vector de síndrome, se verifica que cuanto más baja sea la cantidad de elementos distintos de cero que posea este vector, más cerca se encuentra el candidato de ser una palabra perteneciente al código.

De este modo, la función de ajuste calculada para cada candidato tiene la forma:

$$f_{fitness} = \sum_{i=1}^{v} b_i + \sqrt{\sum_{i-1}^{n} (q_i - \hat{y}_i)^2} \tag{8}$$

Los n elementos componentes del vector q se generan por comparación entre el vector candidato de la población y el vector recibido comprimido $\hat{y}$, de tal manera que:

$$q_i = \begin{cases} 1 & si \quad \hat{y}_i \geq p_i \\ 0 & si \quad \hat{y}_i < p_i \end{cases} \tag{9}$$

La compresión del vector recibido y se debe a que los individuos de la población se generan a partir de números reales uniformemente distribuidos en el intervalo $[0,1]$. Para acomodar los valores recibidos a este intervalo, se calcula primero el mínimo valor entre los recibidos y se toma este valor como extremo izquierdo de una función sigmoidea simétrica respecto de cero. De este modo resulta:

$$\hat{y} = \frac{1}{1 + e^{-a(y-c)}} \tag{10}$$

con $a = mín(y_R)$ y $c = 0$. De este modo, los valores componentes $\hat{y}_i$ quedan confinados proporcionalmente al mismo intervalo numérico que el candidato en cuestión y entonces tiene sentido calcular la distancia entre ambos.
De este modo el segundo término de la función de ajuste resulta proporcional a la distancia entre el vector recibido y el candidato en particular de la población de prueba.
Por su parte, el primer término de la función de ajuste se obtiene calculando el vector síndrome asociado al candidato obtenido por aplicación de la Eq. (9), siendo v la cantidad de componentes del candidato diferentes de cero, lo que en definitiva indica las filas de la matriz H que serán consideradas en el cálculo de síndrome. Este valor también es proporcional a la cercanía de un candidato con una palabra del código.
El cálculo de la función de ajuste se hace para todos los individuos de la población, obteniéndose por cada vez que se corre el algoritmo un candidato óptimo q que es el que mejor se ajusta a la función elegida. De todas maneras, la aplicación del GA se realiza más de una vez, obteniéndose entonces un conjunto de vectores decodificados cuyo número total depende de la cantidad de veces m que se haya seleccionado correr el algoritmo genético. El valor óptimo de m se obtuvo por pruebas, teniendo en cuenta, entre otros parámetros, el tiempo demandado por la decodificación.
Con los m candidatos obtenidos para una palabra decodificada se procede a aplicar el paso siguiente en la decodificación: obtener una única palabra aplicando lógica mayoritaria.

2.3 Aplicación de Lógica Mayoritaria

En este paso se aplica el procedimiento de lógica mayoritaria, muy conocido en el ámbito de la teoría de decodificación para corrección de errores. De este modo se obtiene un único candidato entre todos los posibles obtenidos en el paso previo. Por cada elemento de los candidatos obtenidos se cuentan la cantidad de ceros y unos obtenidos y se elige como resultado para dicho elemento el símbolo con mayor número de elementos. Se debe tener presente que la selección previa tuvo en cuenta la elección de aquellos vectores más cercanos al vector recibido en cada corrida del GA, en total m posibles. De este modo se obtiene el vector decodificado.

2.4 Parámetros elegidos para el decodificador

El parámetro m se obtuvo de manera empírica, teniendo en cuenta los resultados obtenidos en cuanto a la P_{be} para el mismo vector recibido en caso de utilizar la decodificación suma producto y el tiempo demandado por el proceso de decodificación. También se ajustó por tiempo de simulación, cuidando de reducir el mismo lo más posible. Todas las simulaciones se hicieron con Matlab®.
En cuanto al GA, se seleccionó una población original de $P = 600$ individuos tomados a partir de una distribución estocástica uniforme en el intervalo $[0,1]$, eligiéndose un valor de 2 para la cuenta de elite. La fracción de cruce se ajustó en

$P_c = 0.9$. Para la mutación se eligió una función gaussiana, con ajuste de parámetros Escala en 0.5 y Disminución en 0.75 que son los valores por default,. El parámetro Escala determina la varianza de la primera generación, mientras que Disminución controla cómo disminuye la varianza para las siguientes generaciones. Por su parte, el número de Generaciones se limitó a 10, obteniéndose buenos resultados con este criterio de detención.

3 Comparación con Decodificador Tradicional de Suma Producto

El Algoritmo de Decodificación desarrollado en este trabajo es muy diferente al tradicional en cuanto a su forma de operación.

En el algortimo de suma-producto la dimensión de la matriz H y su grado de densidad determinan el grado de complejidad de la decodificación. Siendo n la cantidad de columnas de la matriz H y t el número promedio de unos por columna de esta matriz, el algoritmo suma-producto involucra el cálculo de $6nt$ productos y $5nt$ sumas [3].

En el caso del algoritmo genético, las operaciones dominantes son cruces y permutaciones, en general consideradas operaciones de baja complejidad. Aquí, la verdadera complejidad computacional queda determinada por la elección de la función de ajuste para el problema particular. Esta función determina la capacidad del algoritmo para resolver dicho problema. En la mayoría de los casos prácticos esta función no es sencilla y, generalmente, se requiere alguna clase de modificación del problema para poder encontrar aquella función que mejor lo represente. En el caso de la función seleccionada, en cada iteración, se realizan n sumas, n restas y n productos para obtener el cuadrado de la distancia entre vector recibido y candidato. Luego se calcula la raíz cuadrada del resultado, operación fácilmente realizable mediante una tabla de búsqueda. Recordando que k y n son las longitudes de las palabras mensaje y de las mismas palabras luego de ser codificadas, deducimos, por su parte, la realización del cálculo de síndrome implica una operación de multiplicación de un vector binario de n componentes por una matriz, también binaria, de dimensión $(n-k).n$. Este resultado también se puede realizar sumando las columnas de la matriz H que se correspondan con los elementos no nulos del candidato probado. El valor final que se considera en la función de ajuste es el de la suma de los elementos distintos de cero del síndrome calculado.

Como GA opera de manera aleatoria, no es posible determinar el número de iteraciones que se realizan. En este trabajo los criterios de parada se asocian a la función de error que se trata de minimizar para encontrar el mejor candidato, utilizando un método de búsqueda iterativa dependiente fuertemente de la naturaleza aleatoria del GA.

4 Resultados

En este trabajo se ha evaluado el comportamiento BER de un código LDPC (60, 30) irregular en un canal con ruido multicamino asociado a una función densidad de probabilidad tipo Rayleigh, contrastando el comportamiento del algoritmo de decodificación propuesto con el tradicional de suma-producto [8,15,16].
Al contrario que en el caso tradicional, con el algoritmo de decodificación propuesto no es necesario conocer la dispersión de ruido presente en el canal.
En una primera aproximación, se realizaron simulaciones sobre una muestra de 100 palabras, cada una de $n = 60$ bits, para diferentes valores de contaminación de ruido en el canal, obteniéndose los resultados que se despliegan en las Tablas siguientes.
La Tabla 1 compara la decodificación tradicional de 16 iteraciones con la del algoritmo propuesto ajustado a 15 corridas del GA por palabra. En la Tabla 2 se presenta la misma información pero en el caso de 5 corridas del GA por palabra, con tiempos de procesamiento alrededor de la tercera parte con respecto al caso anterior.

Sigma	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
GA + Lógica Mayoritaria	0	0	0	0	4	10	52	136	212
Suma Producto 16 Iteraciones	0	0	0	0	4	16	61	181	238

Tabla 1. Cantidad total de errores para distintos valores de σ y 15 iteraciones de GA

Sigma	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
GA + Lógica Mayoritaria	0	0	0	0	4	15	59	137	211
Suma Producto 16 Iteraciones	0	0	0	0	4	16	61	181	238

Tabla 2. Cantidad total de errores para distintos valores de σ y 5 iteraciones de GA

Por su parte, la Fig. 1 presenta la comportamiento BER de el decodificador tradicional y el de GA de $m = 5$ iteraciones. Como se puede apreciar en la figura, la curva para el decodificador GA se mantiene en todo el rango de relación señal a ruido por debajo de la del algoritmo tradicional. Se puede concluir se han obtenido valores muy similares en ambos casos.

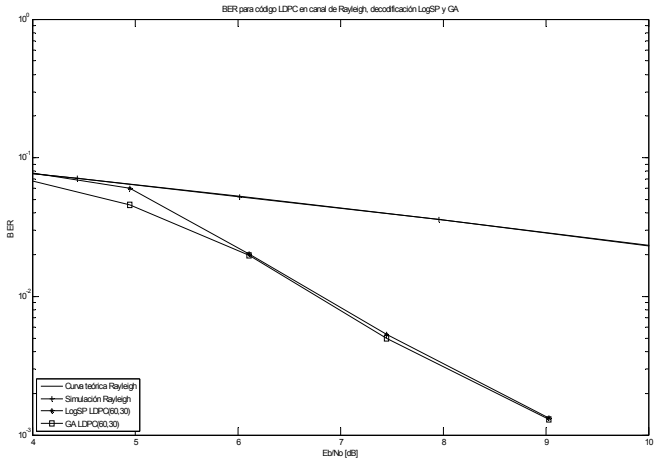

Fig.1. Comportamiento BER de decodificadores, código LDPC(60, 30) irregular .

5 Conclusiones

Se ha probado el decodificador propuesto para varios niveles de ruido y utilizando un código LDPC de tamaño medio. Los resultados de las simulaciones muestran el comportamiento similar de ambos tipos de decodificación. Se resalta la ventaja del decodificador propuesto en cuanto a la ausencia de necesidad de información sobre el nivel de ruido del canal, en contraste con el algoritmo tradicional que necesita tener esta información como parámetro de entrada.
Se propone como trabajo a futuro probar el decodificador propuesto en códigos de mayor longitud y en muestras de mayor cantidad de palabras.

Referencias

1. Subash Shree Pokhrel, and Moon Ho Lee. Comparison of Gaussian Channel with Rayleigh Fading Channel over LDPC-DPC Transmission Scheme Wireless, IET Conference on Mobile and Sensor Networks, Pages: 958 – 961. Date of Current Version: 20 febrero 2009. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4786363
2. Alberto Benvegn`u, Samuele Bandi, Velio Tralli. On the comportamiento and the optimization of LDPC Codes on Fading Channels with Imperfect CSI. IEEE International Symposium on Wireless Communication Systems. 2008. Pages: 539 - 541 Date of Current Version: 30 diciembre 2008. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4726114
3. Leonardo Jos¶e Arnone. Transmisión Segura en Comunicaciones Inlámbricas de Corto Alcance. Tesis presentada para optar por el Grado Académico de Doctor en Ingeniería, mención Electrónica. Director de Tesis: Dr. Jorge Castiñeira Moreira. Mayo de 2008

4. A. G. Scandurra, A. L. D. Pra, L. Arnone, L. Passoni, and J. C.Moreira, “A genetic-algorithm based decoder for low density parity check codes,” Latin American Applied Research, vol. 36, pp. 169–172, Sept. 2006.
5. Bonissone P., "Soft Computing and Meta-heuristics: using knowledge and reasoning to control search and vice-versa,” Proc. SPIE Applications and Science of Neural Networks, Fuzzy Systems and Evolutionary Computation V, S. Diego, CA, 133--149, (2003).
6. Daijin K., and A. Sunha, “A MS-GS VQ Codebook Design for Wireless Image Communication Using Genetic Algorithms,” IEEE Trans. Evol. Comput., 3, 35-52 (1999).
7. Dontas K. and K. De Jong, “Discovery of Maximal Distance Codes Using Genetic Algorithms,” Proc. of the 2nd International IEEE Conference on Tools for Artificial Intelligence, Herndon, VA, 805-811, (1990).
8. Gallager R.G., Low-Density Parity-Check Codes, MIT Press, Cambridge, MA, (1963).
9. Goldberg, D.E., Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, New York, (1989).
10. Koza, J.R., Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, MA, (1992).
11. MacKay D.J.C, and R.M. Neal, “Near Shannon Limit Performance of Low-Density Parity-check Codes,” IEE Elect. Lett., 33 , 457-458, (1997).
12. MacKay D.J.C. and C.P. Hesketh, “Comportamiento of low density parity check codes as a function of actual and assumed noise levels,” Electronic Notes in Theoretical Computer Science, 74, 1-8, (2003).
13. MacKay D.J.C., “Good error-correcting codes based on very sparse matrices," IEEE Trans. Inform. Theory, 45, 399-431, (1999).
14. Michalewicz, Z., Genetic algorithms + data structures = evolution programs, Springer-Verlag, Berlin, (1992).
15. Richardson T., A. Shokrollahi and R. Urbanke, “Design of capacity approaching irregular Low-Density Parity-Check codes,” IEEE Trans. Inform. Theory, 47, 619-637, (2001).
16. Richardson T. and R. Urbanke, “The capacity of Low-Density Parity-Check codes under message-passing decoding,” IEEE Trans. Inform. Theory, 47, 599-618, (2001).
17. Schaffer J., J. Caruana, L. Eshelman and R. Das, “A study of control parameters affecting online comportamiento of genetic algorithms for function optimization,” Proc. of the Third international Conference on Genetic Algorithms, San Mateo, CA, 51-60, (1989).
18. Valenzuela, C.L. and P.Y Wang, “VLSI placement and area optimization using a genetic algorithm to breed normalized postfix expressions,” IEEE Trans. Evol. Comput., 6, 390-401, (2002).