*Monografías de la Real Academia de Ciencias. Zaragoza.* **43:** 99–102, (2018).
ISSN: 1132-6360

# CÁLCULO APROXIMADO DE LA DISTANCIA MÍNIMA DE UN CÓDIGO LINEAL

JOSÉ GÓMEZ-TORRECILLAS, F. J. LOBILLO Y GABRIEL NAVARRO

Resumen. Mostramos una aplicación de los algoritmos aproximados al problema NP-duro del cálculo de la distancia mínima de un código lineal sobre un cuerpo finito. En particular, desarrollamos un algoritmo genético para obtener una cota superior. A diferencia de los algoritmos (exactos o aproximados) presentes en la literatura, la eficiencia teórica no es polinómica solamente respecto de la longitud del código.

## Introducción

A la hora de valorar la utilidad práctica de un código lineal, la distancia mínima es uno de los parámetros principales, ya que permite calcular la capacidad de corrección de dicho código. Entre los algoritmos desarrollados para su cálculo, el más rápido es el algoritmo de Brouwer-Zimmermann (BZ), ver [2]. Aunque el algoritmo de BZ puede aplicarse a códigos sobre cualquier cuerpo finito, en la práctica sólo puede ser considerado eficaz para códigos binarios. Sin embargo, existen situaciones donde es necesario trabajar sobre códigos no binarios. Por ejemplo, en el diseño de códigos cíclicos torcidos [5], tanto de bloque como convolucionales, y sus algoritmos de decodificación [6, 8] se necesita que el cuerpo finito base tenga un cardinal relativamente grande (ver los ejemplos de [7]). Esto es así puesto que, a mayor longitud, es necesario aumentar el cardinal del cuerpo.

Es conocido que el cálculo de la distancia es un problema NP-duro, y su problema de decisión asociado, NP-completo [10]. Así, salvo que P=NP, no puede existir un algoritmo (exacto) que calcule la distancia de cualquier código lineal en un tiempo razonable. Aunque en la literatura pueden encontrarse algoritmos aproximados, como, por ejemplo, en [1] o en [9], todos estos métodos siguen considerando un espacio de soluciones que crece de forma exponencial respecto al cardinal del subcuerpo primo así como respecto de la dimensión del código. Nuestra propuesta consiste en el diseño, e implementación, de un algoritmo aproximado cuyo espacio de de soluciones y tiempo de ejecución únicamente depende no polinomialmente de la longitud del código. Concretamente, diseñamos un algoritmo genético, basado en un modelo generacional clásico, para el cálculo de una cota superior de la distancia.

## 1. Algoritmo genético GLN

Los *algoritmos genéticos* son un tipo de metaheurísticas que siguen un modelo de búsqueda basado en poblaciones consistente en simular el proceso de combinación genética. Consultar [4, Capítulo 3] para una referencia básica. A continuación detallamos los módulos que componen el Algoritmo 2.

**1.1. Representación del espacio de búsqueda.** El desarrollo de un algoritmo genético requiere, en primer lugar, representar el espacio de soluciones del problema mediante cierta codificación (los *cromosomas*) de manera que, por *recombinación* de ellos y *mutación*, nos aproximemos a algún cromosoma óptimo. Para el problema de la distancia mínima, una representación obvia es considerar una extensión de la utilizada en [1] para códigos binarios, de modo que el espacio de soluciones sea un $\mathbb{F}_q$–espacio vectorial de dimensión $k$. Sin embargo, con esta representación, el tamaño de dicho espacio aumenta de forma exponencial respecto del tamaño del subcuerpo primo (así como de $k$). Nuestra propuesta considera un espacio de representación que no depende del cuerpo. Para ello, necesitamos el siguiente resultado. Aunque su demostración no es complicada, no lo hemos encontrado en la literatura.

**Teorema 1.1.** *Sea $G$ una matriz $k \times n$ generadora de un código $[n,k]$-lineal $\mathcal{C}$ sobre el cuerpo finito $\mathbb{F}_q$. Existe una permutación $P \in \mathcal{S}_n$ tal que la matriz reducida por filas $R$ de $GM_P$, donde $M_P$ es la matriz asociada a $P$, verifica que el peso de Hamming de alguna de sus filas coincide con la distancia mínima de $\mathcal{C}$. Además, si $b$ es la fila de $R$ verificando tal propiedad, entonces $bM_P^{-1}$ es una palabra de $\mathcal{C}$ de peso mínimo.*

Por tanto, el problema de calcular la distancia mínima de un código lineal se reduce a calcular el mínimo de la aplicación $\mathfrak{d} : \mathcal{S}_n \to \mathbb{N}$ definida como

$$\mathfrak{d}(P) = \min\{\mathrm{w}(b) \mid b \text{ es una fila de la matriz reducida por filas de } GM_P\},$$

donde $\mathrm{w}(b)$ denota el peso de Hamming del vector $b$. Nótese que con esta representación el espacio de soluciones sólo depende de la longitud del código. En particular, es invariante respecto al tamaño del cuerpo. Obviamente, el cálculo de la imagen de una permutación por $\mathfrak{d}$ sí depende del cuerpo y de la dimensión del código. Sin embargo, la eficiencia de este cálculo respecto del cuerpo primo y la dimensión es polinomial.

**1.2. Operador de cruce.** La recombinación de cromosomas es uno de los puntos clave para conseguir un equilibrio adecuado entre *diversidad* (exploración de diferentes zonas del espacio de búsqueda) y *convergencia* en el comportamiento del algoritmo. Los operadores de cruce clásicos no consideran la estructura algebraica del grupo $\mathcal{S}_n$. En esta comunicación proponemos basar el operador de cruce en el producto de permutaciones, ya que dos (o más) elementos al azar de $\mathcal{S}_n$ probablemente formen un sistema de generadores (Teorema de Dickson [3, Theorem 1]) y, por tanto, podemos esperar una diversidad suficiente para obtener un cromosoma óptimo. Concretamente, definimos una familia de cruces algebraicos $AX_r$, donde $r$ es el número de cromosomas involucrados. Para $r$ cromosomas $p_1, p_2, \ldots, p_r$ consideramos

$$\mathcal{T} = \{p_{\tau(1)} \circ p_{\tau(2)} \circ \cdots \circ p_{\tau(r)} \text{ tal que } \tau \in \mathcal{S}_r\}.$$

De esta familia seleccionamos los $r$ cromosomas con menor imagen por $\mathfrak{d}$ (aquellos que presentan una mejor adaptación) que reemplazarán a los cromosomas originales. Mantendremos ciertos conjuntos de individuos sin cruce utilizando una probabilidad de cruce $p_c$. En la Sección 2, utilizaremos $p_c = 0{,}7$, que es el valor más común en la literatura.

**1.3. Operador de mutación.** Este operador aumenta la diversidad al permitir la mutación de algunos genes (componentes de un cromosoma). El operador de mutación estándar consiste en la multiplicación por una transposición aleatoria. Sin embargo, en este problema,

la mutación por una transposición no suele cambiar el valor de $\mathfrak{d}$, por lo que utilizaremos ciclos de longitud $\lfloor n/2 \rfloor$. La probabilidad de mutación utilizada en la Sección 2 es $p_m = 0{,}1$.

**1.4. Relevo generacional.** Con las características anteriores, podemos diseñar el Algoritmo 1 para el cálculo de una población a partir otra existente. Está basado en un modelo generacional básico: la nueva población reemplaza a la antigua. Para aumentar la convergencia, la población resultante hereda el cromosoma con mejor adaptación (esto recibe el nombre de elitismo).

**Algoritmo 1** RelevoGeneracional

**Entrada:** $P(i)$, población en el instante $i \geq 0$; $G$, matriz generadora; $p_c$, probabilidad de cruce; $p_m$, probabilidad de mutación; $r$, cardinal del conjunto susceptible de cruzarse.
**Salida:** $P(i+1)$ población en el instante $i+1$.

$best \leftarrow$ cromosoma donde se alcanza el mínimo de $\mathfrak{d}$ para $P(i)$.
$X \leftarrow \emptyset$
**Mientras** $\#P(i) \geq r$ **hacer**
  $\mathcal{S} \leftarrow \{r$ cromosomas elegidos aleatoriamente de $P(i)\}$.
  $P(i) \leftarrow P(i) - \mathcal{S}$
  $\mathcal{S} \leftarrow \begin{cases} AX_r(\mathcal{S}) \text{ con probabilidad } p_c, \\ \mathcal{S} \text{ con probabilidad } 1 - p_c. \end{cases}$
  **Para** $s \in \mathcal{S}$ **hacer**
    $s \leftarrow \begin{cases} s \circ p \text{ con probabilidad } p_m,\ p \text{ ciclo aleatorio de longitud } \lfloor n/2 \rfloor \\ s \text{ con probabilidad } 1 - p_m. \end{cases}$
  $X \leftarrow X \cup \mathcal{S}$
**Devolver** $X \cup \{best\}$

**1.5. Población inicial.** Como población inicial utilizaremos $2n$ permutaciones elegidas aleatoriamente. Puesto que cualquier permutación se obtiene como un producto de los elementos de un sistema de generadores, buscamos que la población activa probablemente forme un sistema de generadores de $\mathcal{S}_n$. El Algoritmo 2 recoge todos los módulos explicados en esta sección.

**Algoritmo 2** Algoritmo genético GLN

**Entrada:** $(G, r, p_m, p_c)$, como en Algoritmo 1; $c$ ó $t$, número máximo de iteraciones ó tiempo máximo de ejecución, respectivamente.
**Salida:** $\overline{d}$ cota superior de la distancia mínima de $\mathcal{C}$.

$P(0) \leftarrow$ Población inicial aleatoria de tamaño $2n$.
$i \leftarrow 1$
**Mientras** $i < c$ (ó $time < t$) **hacer**
  $P(i) \leftarrow \text{RelevoGeneracional}(P(i-1), G, r, p_c, p_m)$
  $i \leftarrow i + 1$
$best \leftarrow$ cromosoma donde se alcanza el mínimo de $\mathfrak{d}$ para $P(i)$.
**Devolver** $\mathfrak{d}(best)$

## 2. Resultados experimentales

En el Cuadro 1 mostramos los resultados experimentales preliminares al aplicar el Algoritmo 2 a algunos de los códigos lineales de la base de datos de `http://codetables.de`. Las matrices generadoras correspondientes han sido calculadas con MAGMA usando la función BKLC. El Algoritmo 2 ha sido implementado en Sagemath (Python). La ejecución del algoritmo se ha realizado usando un procesador Intel Core i7 3GHz bajo el sistema operativo macOS 10.12.6. Como comparación, una implementación del algoritmo de BZ requiere un tiempo de ejecución estimado de más de 40 horas para el código $[30, 12, 14]_8$-lineal

| $n$ | $k$ | $d$ | tiempo medio (seg.) | $n$ | $k$ | $d$ | tiempo medio (seg.) |
|---|---|---|---|---|---|---|---|
| 30 | 12 | 14 | 0.15 | 60 | 25 | 24 | 0.38 |
| 30 | 14 | 12 | 0.16 | 60 | 33 | 18 | 0.40 |
| 30 | 16 | 10 | 0.16 | 60 | 41 | 12 | 0.43 |
| 45 | 18 | 19 | 0.26 | 75 | 25 | 33 | 5.43 |
| 45 | 21 | 16 | 0.69 | 75 | 35 | 24 | 3.31 |
| 45 | 25 | 13 | 0.27 | 75 | 45 | 17 | 10.15 |

Cuadro 1. Tiempo medio (100 repeticiones del Algoritmo 2 con $AX_2$) para alcanzar la distancia mínima en algunos códigos $[n, k, d]$-lineales sobre $\mathbb{F}_8$

**Investigación financiada por la ayuda MTM2016-78364-P de la Agencia Estatal de Investigación y FEDER.**

## Referencias

[1] M. Askali, A. Azouaoui, S. Nouh, M. Belkasmi. On the computing of the minimum distance of linear block codes by heuristic methods, International Journal of Communications, Network and System Sciences 5 (11) (2012), 774–784.

[2] A. Betten, M. Braun, H. Fripertinger, A. Kerber, A. Kohnert, A. Wassermann. Error-Correcting Linear Codes. Algorithms and Computation in Mathematics 18. Springer. 2006.

[3] J. D. Dixon, The probability of generating the symmetric group. Mathematische Zeitschrift, 110 (1969), 199–205.

[4] E-G. Talbi. Metaheuristics: From Design to Implementation. John Wiley & Sons, Inc. 2009.

[5] J. Gómez-Torrecillas, F.J. Lobillo, G. Navarro. A new perspective of cyclicity in convolutional codes, IEEE Transactions on Information Theory 62 (5) (2016), 2702–2706.

[6] J. Gómez-Torrecillas, F.J. Lobillo, G. Navarro. A Sugiyama-like decoding algorithm for convolutional codes, IEEE Transactions on Information Theory 63 (2017) 6216–6226.

[7] J. Gómez-Torrecillas, F.J. Lobillo G. Navarro A. Neri. Hartmann-Tzeng bound and skew cyclic codes of designed Hamming distance, Finite Fields and Their Applications 50 (2018), 84–112.

[8] J. Gómez-Torrecillas, F.J. Lobillo, G. Navarro. Peterson-Gorenstein-Zierler algorithm for skew RS codes, Linear and Multilinear Algebra 66 (2018), 469–487.

[9] J. S. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes, IEEE Transactions on Information Theory 34 (5) (1988), 1354–1359.

[10] A. Vardy. The intractability of computing the minimum distance of a code, IEEE Transactions on Information Theory 43 (6) (1997), 1757–1766.

Departamento de Álgebra y CITIC, Universidad de Granada
*E-mail address*: `gomezj@ugr.es`
*E-mail address*: `jlobillo@ugr.es`

Departamento de Ciencias de la Computación e IA y CITIC, Universidad de Granada
*E-mail address*: `gnavarro@ugr.es`