

Ampliació d'una llibreria en MAGMA per a codis Z_2Z_4 -lineals

Cristina Diéguez

Resum — Un codi Z_2Z_4 -lineal és un codi C format per un conjunt de α coordenades binàries i β coordenades quaternàries. Es creen una sèrie de funcions sobre aquest anell $Z_2^\alpha \times Z_4^\beta$ per calcular, en MAGMA, el pes mínim de Lee, la distància mínima de Lee, el nombre mínim de paraules-codi i la distribució de pesos usant l'algorisme basat en el còmput del kernel i l'algorisme basat en la força bruta. Posteriorment, es presenta una comparativa de rendiment entre aquests dos algorismes utilitzant dues famílies de codis diferents per tal de poder determinar sota quines característiques quins dels dos algorismes té uns temps de còmput inferiors.

Paraules clau — Codis Z_2Z_4 -lineals, Codis Z_2Z_4 -additius, MAGMA, Pes mínim de Lee, Distància mínima de Lee, Nombre mínim de paraules-codi, Distribució de pesos, Algorisme basat en el kernel, Algorisme basat en la força bruta.

Abstract — A Z_2Z_4 -linear code is a code C determined by α binary coordinates and β quaternary coordinates. A set of functions on this $Z_2^\alpha \times Z_4^\beta$ ring is presented for computing, in MAGMA, minimum Lee weight, minimum Lee distance, minimum codewords and weight distribution using a kernel-based algorithm and an algorithm based on brute force. Then, a performance comparison, which uses two different code families, is presented in order to conclude what characteristics involve a lower calculation time of both algorithms mentioned.

Index Terms — Z_2Z_4 -linear codes, Z_2Z_4 -additive codes, MAGMA, Minimum Lee weight, Minimum Lee distance, Minimum codewords, Weight distribution, Kernel-based algorithm, BruteForce algorithm.



1 INTRODUCCIÓ

OT procés d'enviament i recepció de missatges es realitza a través d'un canal de comunicació. No obstant, el missatge rebut a la sortida del canal pot haver patit alguna alteració a causa del soroll i fa que aquest es converteixi en erroni. Aleshores, perquè el receptor pugui eliminar possibles errors es realitza el procés de codificació-descodificació.

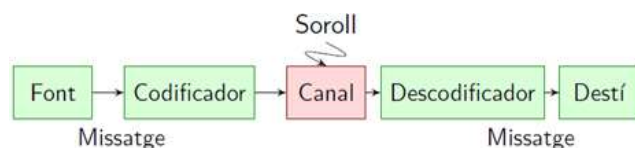


Figura 1. Procés de comunicació.

En la codificació, s'assigna a cada símbol o conjunt de símbols de la font una paraula-codi afegint redundància. L'agrupació de totes aquestes paraules-codi formen un codi.

En el procés de descodificació, si s'ha produït algun error en la transmissió i el descodificador detecta l'error, es realitza la correcció aprofitant la redundància afegida. Aleshores el que es vol és poder seguir enviant missatges tenint la capacitat de detecció i correcció d'errors. El segon teorema de Shannon especifica aquest fet ja que declara que és possible transmetre informació a través d'un canal amb soroll si es transmet a una taxa de transmissió

de la informació¹ inferior a la capacitat del canal. Aquest teorema dona peu a la teoria de codificació per a la correcció d'errors que té per objectiu la construcció de codificadors i descodificadors que permetin enviar el màxim d'informació amb el mínim d'errors.

L'escenari ideal seria trobar codis on cadascun dels vectors tingués una única descodificació possible, però trobar codis amb aquestes propietats no és una tasca banal. Una manera de poder minimitzar els errors seria augmentant la redundància en el missatge per poder, després, corregir els errors. El desavantatge que presenta és que existeix una penalització de la velocitat de transmissió.

Dins de l'àmbit informàtic, el tipus de codis que presenten un gran interès per les seves característiques són els codis lineals sobre anells perquè experimenten les següents propietats:

- Tancament per la suma: dues paraules codi sumades és igual a una altra paraula-codi.
- Tancament per la multiplicació d'un escalar: si es multiplica una paraula-codi per un escalar, dona una altra paraula-codi.

Gràcies a aquestes propietats, es pot trobar un conjunt mínim de paraules-codis linealment independents

¹ Taxa de transmissió de la informació $R_I = \frac{\log_2 |C|}{n}$

que són capaces de generar totes les paraules del codi. Aleshores, s'observa que un codi pot ser representat de forma matricial per les paraules-codi linealment independents. Es crea el que s'anomena matriu generadora que facilita el treball i la representació d'aquest tipus de codis. Amb aquest fet, es redueixen els costos de memòria i còmput a l'hora de treballar amb codis.

A part d'usar codis binaris que són un tipus de representació utilitzant l'anell d'enters mòdul 2, actualment es treballa amb codis Z_2Z_4 -additius, que són codis sobre l'anell $Z_2^\alpha \times Z_4^\beta$. Aquests codis també posseeixen el concepte de matriu generadora G que està formada per:

- γ vectors d'ordre 2: aquells formats per coordenades a $\{0,2\}$.
- δ vectors d'ordre 4: aquells que contenen alguna coordenada a $\{1,3\}$.

Coneixent aquesta informació, es determina que el codi C obtingut a partir de la matriu generadora G és d'ordre $2^\gamma \cdot 4^\delta$. Aquesta expressió, a la vegada, proporciona el número de paraules-codi de C . A l'hora de generar-les, s'opera $(x,y) \cdot G$ on x pertany a Z_2^γ i y a Z_4^δ ja que si x correspongués a Z_4 es duplicarien les mateixes paraules-codi.

Amb la finalitat de poder treballar a Z_2 , existeix una funció anomenada *Gray map* ϕ que proporciona les imatges de les paraules-codi en aquest conjunt; és a dir, realitza una bijecció:

$$\begin{array}{ccc} \phi: Z_4 & \longrightarrow & Z_2 \\ 0 & \longrightarrow & 00 \\ 1 & \longrightarrow & 01 \\ 2 & \longrightarrow & 11 \\ 3 & \longrightarrow & 10 \end{array}$$

Si $v=(v1,v2)$ que pertany a $Z_2^\alpha \times Z_4^\beta$, es defineix $\phi(v)=(v1, \phi(v2))$. Aquesta bijecció només afecta, realment, a les β coordenades quaternàries de les paraules-codi mentre que les α coordenades binàries es mantenen iguals. El codi binari $\phi(C)$ s'anomena Z_2Z_4 - lineal i, de forma general, no és lineal.

La distància de Hamming $d_H(u,v)$ entre dues paraules-codi u,v que pertanyen a Z_2 és el nombre de coordenades per les quals difereixen u i v . El pes de Hamming $w_H(u)$ és la distància de Hamming entre la paraula-codi u i la paraula-codi formada per un vector de tots 0. En codis quaternaris, que són un tipus de representació utilitzant l'anell d'enters mòdul 4, es fa servir la mètrica de Lee. El pes de Lee $w_L(v)$ és la suma dels pesos de Lee de les seves coordenades. Els pesos d'aquestes coordenades estan definits de la següent manera:

- $w_L(0)=0$
- $w_L(1)=w_L(3)=1$
- $w_L(2)=2$

La funció *Gray map* ϕ preserva les distàncies, transformant la distància de Lee a distància de Hamming. Gràcies a aquesta funció, s'observa que el pes de Lee d'un vector, v , sobre Z_4 coincideix amb el pes de Hamming de $\phi(v)$ sobre Z_2 . Un exemple d'aquest succés seria:

$$\phi(10230) = (0100111000)$$

$$w_L(10230) = w_L(1) + w_L(0) + w_L(2) + w_L(3) + w_L(0) = 4$$

$$w_H(0100111000) = 4$$

La distància mínima d'un codi C és el mínim valor de la distància de totes les paraules-codi que formen C . Així mateix, el pes mínim d'un codi C és el mínim valor dels pesos de totes les paraules-codi. En aquest projecte, es va treballar amb aquests conceptes a l'hora de plantejar i implementar funcions que ajudessin en el procés de descodificació, la qual pot ser descodificació via síndrome o descodificació per mínima distància.

2 ESTAT DE L'ART

El grup CCSG és un equip format dins del Departament d'Enginyeria de la Informació i de les Comunicacions, d'EIC, i dedicat, parcialment, a la investigació de codis Z_2Z_4 -lineals amb la finalitat dur a terme els següents objectius, entre d'altres:

- Construir i caracteritzar nous codis Z_2Z_4 -lineals, i computar els seus paràmetres estructurals.
- Caracteritzar i construir famílies de codis Reed-Muller Z_2Z_4 -lineals, i computar-ne el rang i la dimensió del kernel.
- Desenvolupar els algorismes necessaris per establir noves opcions en l'ocultació de dades i en l'autenticació de documents, utilitzant codis Z_2Z_4 -lineals.
- Expandir el software de MAGMA implementant nous paquets amb la finalitat de treballar eficientment amb codis Z_2Z_4 -lineals, codis regulars i codis no lineals.

Un dels principals articles escrits sobre codis Z_2Z_4 -lineals és [1] en què s'estudien els codis Z_2Z_4 -additius a partir dels quals s'extreu la corresponent imatge binària i s'obtenen els codis Z_2Z_4 -lineals. D'aquests nous codis, es detallen les seves propietats i les formes estàndard de les matrius generadores i de control. En aquest article se'n fa la primera definició. També esmentar l'article [4] ja que s'hi demostren les propietats necessàries pel càlcul de rangs i kernels dels codis Z_2Z_4 -lineals.

Degut que en aquest projecte ha estat necessari

treballar amb les distàncies de Hamming dels codis ja mencionats, una de les referències a destacar és [6] perquè defineix i implementa mètodes algorítmics per la computació de la distància mínima de Hamming per codis Z_2Z_4 -lineals.

El llenguatge MAGMA porta incorporat llibreries sobre Z_2 i sobre Z_4 cosa que facilita el treball sobre aquests anells. Gràcies als membres d'aquest grup d'investigació CCSG, des de fa uns anys, es duu a terme el desenvolupament d'una nova llibreria en MAGMA per permetre treballar amb codis Z_2Z_4 -lineals, fent ús d'algunes llibreries ja existents en aquest llenguatge. Es pretén que aquesta estigui totalment integrada amb la llibreria per a codis lineals.

Com s'ha esmentat anteriorment, s'ha usat el llenguatge MAGMA amb la finalitat de realitzar la part pràctica del projecte. Gràcies a l'estructura Z_2Z_4 -lineals es pot treballar amb aquest llenguatge de manera lineal a $Z_2^\alpha \times Z_4^\beta$ i també es poden relacionar aquests codis amb codis sobre Z_2 i sobre Z_4 .

Les referències existents sobre l'ús d'aquest llenguatge en l'entorn desitjat són les proporcionades per MAGMA, [9] i [10]. Addicionalment, també es van consultar les guies que facilita el CCSG per MAGMA [8] i per entendre els paquets de codis Z_2Z_4 -lineals, [2], i codis Z_2Z_4 -additius, [3].

El fet d'haver de dissenyar i implementar unes funcions que s'integraran dins de MAGMA implica que s'han de seguir els estàndards marcats per tal mantenir el format. Com a conseqüència, es consultarà la guia [7] ja que proporciona aquests estàndards.

3 OBJECTIUS

Els objectius que es presenten en aquest article són els proposats a l'inici del projecte i, que al mateix temps, coincideixen amb els escollits durant el transcurs del treball.

- Realització d'un estudi dels codis Z_2Z_4 -lineals tant a nivell conceptual com de les propietats que els defineixen.
- Aprenentatge d'un nou llenguatge i entorn de programació anomenat MAGMA.
- Disseny i desenvolupament de tests de prova i tests de rendiment de les funcions realitzades prèviament.
- Creació de noves funcions de codis Z_2Z_4 -lineals per realitzar els càlcul del pes mínim de Lee, la distància mínima de Lee, el nombre mínim de paraules-codi i la distribució de pesos.
- Ampliació de la llibreria de MAGMA existent sobre teoria de codis Z_2Z_4 -lineals amb les noves funcions desenvolupades per tal de facilitar la descodificació via síndrome.

4 METODOLOGIA I MATERIAL

4.1 Metodologia

Aquest projecte va constar d'un component teòric rellevant a partir del qual es van poder desenvolupar les funcions necessàries per aquesta ampliació de la llibreria de MAGMA. L'expansió de coneixement esmentada va implicar un fil de continuïtat constant al llarg de tot el projecte ja que era necessari aquest suport a l'hora de tractar els codis Z_2Z_4 -lineals. Com a conseqüència, aquesta obtenció de coneixement va provenir de les publicacions al respecte, per exemple [1,4,5,6], i de l'ajuda i assessorament de la tutora del projecte Cristina Fernández. També es van seguir els formats establerts per codis lineals binaris i quaternaris referenciats en els articles consultats.

El component pràctic va adoptar la metodologia de treball de MAGMA sobre codis lineals en els diferents anells aprofitant les llibreries que aquest ja presenta. La seva codificació va seguir l'estil estrictament marcat pels estàndards facilitant, així, la seva integració a la llibreria. Les funcions previstes a desenvolupar durant tota el procés són les mostrades a continuació i cadascuna d'elles es va implementar amb l'algorisme basat en el càlcul del kernel i l'algorismes basat en la força bruta.

- `Z2Z4MinimumLeeWeight`
- `Z2Z4MinimumLeeDistance`
- `Z2Z4MinimumWord`
- `Z2Z4MinimumWords`
- `WeightDistribution`

El procés de desenvolupament del projecte va seguir una pràctica de programació d'enginyeria de software anomenada *Test-Driven Development (TDD)*. Aquesta es basa en la idea d'elaborar primer les proves, després escriure el codi font que serà avaluat amb els tests fets i, per últim, fer la reestructuració o refacció del codi escrit. Al realitzar la metodologia en aquest ordre descrit en la figura 2, va obligar a fer un exercici previ d'anàlisi dels requisits i dels diversos escenaris possibles. També implicava una reducció del codi al mínim imprescindible per resoldre la prova que hi tenia associada minimitzant errors. Degut a l'important paper que jugaven els test, va ser imprescindible la realització d'un conjunt de proves unitàries que cobrissin tots els casos a avaluar. Al mateix temps, l'etapa de reestructuració també era rellevant ja que facilita la comprensió del codi i millora l'estructura interna, sense canviar el comportament extern. Amb aquesta pràctica s'aconsegueixen codis més robusts, més segurs, més llegibles, amb més facilitat per mantenir-los al llarg del temps i una major rapidesa en el desenvolupament.

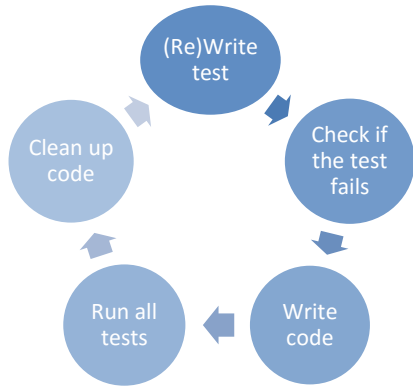


Figura 2: Cicle de vida de Test-Driven Development.

En la fase d'elaboració de proves, es va dur a terme la descripció i implementació de tests de caixa negra programats on s'especificaven quins eren els resultats correctes desitjats per poder realitzar la comparació amb els obtinguts. D'aquesta forma es van poder detectar i observar errors en les funcions programades amb l'objectiu de modificar-les i reavaluar-les.

Conseqüentment, per cada tipus de codi, es van seguir els passos esmentats seguidament per tal realitzar la verificació de les funcions a testejar:

- Generar el codi amb les característiques específiques.
- Determinar els valors esperats dels paràmetres a provar.
- Comprovar que aquests valors previstos coincideixin amb els resultats que genera cada funció testejada.

Un cop aquesta etapa es va acabar, va començar la fase d'experimentació amb la qual es van dissenyar una sèrie de codis per poder testejar el rendiment de les funcions segons els dos tipus d'algorismes implementats. D'aquesta forma es va poder avaluar el comportament de cada algorisme segons les característiques que presentessin cada família de codis i poder triar quin dels dos era més òptim o si era necessari usar els dos, quin escollir sota circumstàncies determinades.

4.2 Material

La fase d'experimentació d'aquest projecte va utilitzar dues famílies de codis diferents per poder estudiar el rendiment de les funcions definides en els objectius.

La primera va ser els codis de Hadamard. En aquests codis, es van definir diversos valors de m i δ modificant, així, la dimensió del kernel i el nombre de cosets representatius. Tot i que aquests codis són codis sobre Z_4 , es van generar sobre l'anell $Z_2^\alpha \times Z_4^\beta$.

Hadamard Codes (δ, m)	Kernel Dimension	Number Cosets Representatives
HadamardCode (3, 10)	9	3
HadamardCode (4, 10)	8	7
HadamardCode (5, 10)	7	15
HadamardCode (3, 11)	10	2
HadamardCode (4, 11)	9	7
HadamardCode (5, 11)	8	15
HadamardCode (6, 11)	7	31

L'altra família de codis utilitzats es van obtenir a partir de les matrius mostrades a l'apèndix A.1 i generats sobre l'anell $Z_2^\alpha \times Z_4^\beta$. Aquests codis conserven la mateixa longitud i el nombre de paraules-codi, però canvia la dimensió del kernel.

Z_2Z_4 Codes	Kernel Dimension	Number Cosets Representatives
C1	16	0
C2	15	3
C3	13	7
C4	12	15
C5	11	31
C6	10	63

Amb aquestes dues famílies de codis es van dissenyar els tests de rendiments que calculaven el temps empleat per cadascuna de les funcions desenvolupades amb els dos algorismes esmentats: el basat amb el càlcul del kernel i el de la força bruta.

5 RESULTATS

Els resultats, que es van obtenir després de l'execució dels tests de rendiment sobre cadascuna de les funcions implementades, es diferencien segons la família de codis emprada.

Per cada família de codis, es mostren els gràfics dels temps de còmput de cadascuna de les funcions distingint els dos algorismes: el basat en el càlcul del kernel i el basat en la força bruta. Al mateix temps, també es presenta el temps de càlcul del kernel per cadascuna.

5.1 Codis de Hadamard

Minimum Lee Weight

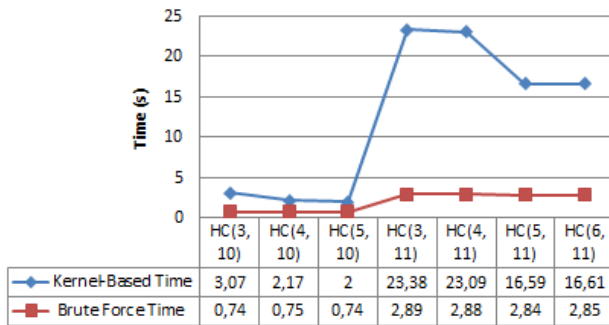


Figura 3. Temps de càlcul pel pes mínim de Lee per codis Hadamard.

Minimum Lee Distance

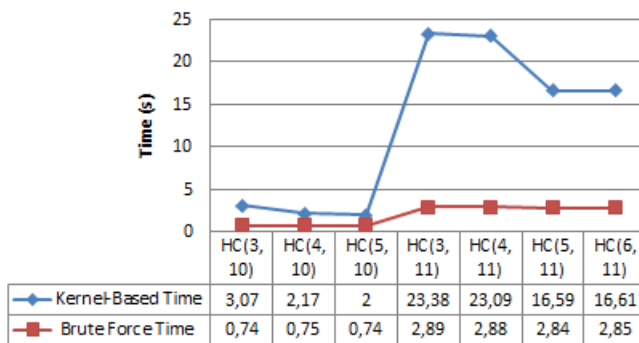


Figura 4. Temps de càlcul per la distància mínima de Lee per codis Hadamard.

Minimum Word

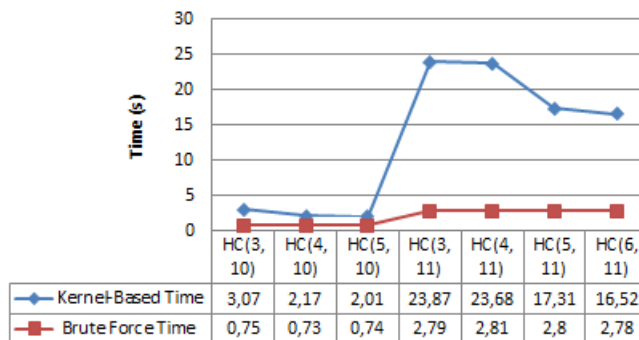


Figura 5. Temps de càlcul per la paraula-codi de pes mínim per codis Hadamard.

Minimum Words

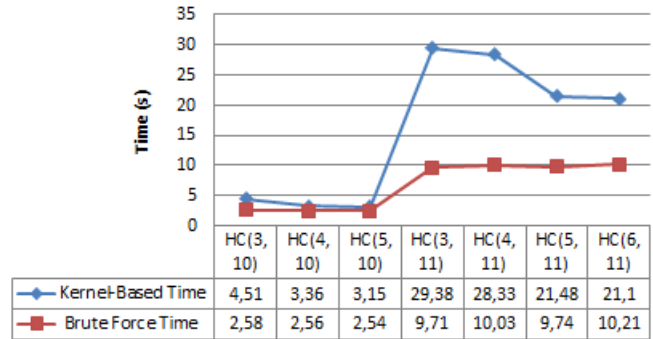


Figura 6. Temps de càlcul per les paraules-codi de pes mínim per codis Hadamard.

Kernel Time

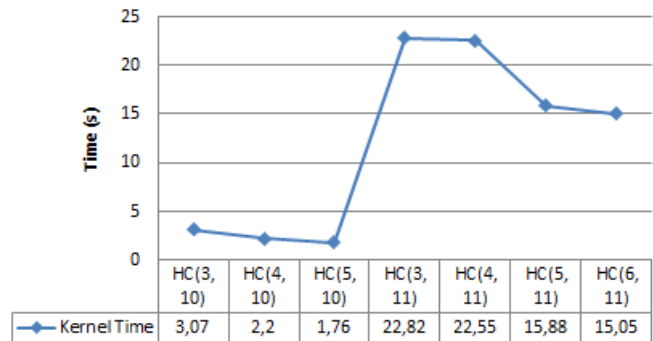


Figura 7. Temps de càlcul del Kernel per codis Hadamard.

Amb aquesta família de codis s'observa que l'algorisme més eficient pel càlcul de les diverses funcions és el basat en la força bruta ja que els seus temps són inferiors en tots els casos.

En tots els gràfics mostrats, es detecta un punt a partir del qual els dos algorismes es distancien de forma desmesurada. L'algorisme basat en el càlcul del kernel augmenta els seus temps en totes les funcions al voltant de la vintena, mentre que en l'algorisme basat en la força bruta els seus temps no sobrepassen els 3 segons. Aquest punt de divergència es produeix quan el paràmetre m dels codis de Hadamard és 11.

L'augment del paràmetre δ no és rellevant en el cas dels codis de Hadamard $m=10$, però en els codis de Hadamard on el paràmetre $m=11$ sí que s'observa una disminució dels temps de càlcul quan aquest δ incrementa.

5.2 Codis Z_2Z_4

Minimum Lee Weight

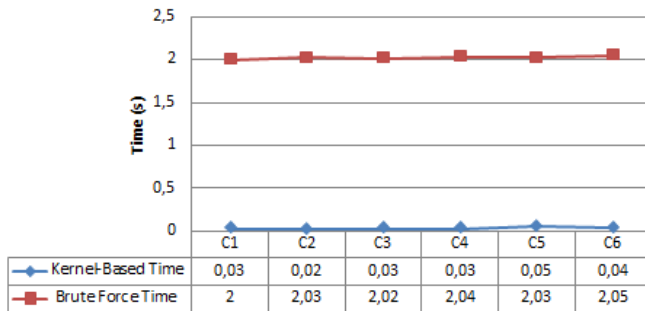


Figura 8. Temps de càlcul pel pes mínim de Lee per codis Z_2Z_4 .

Minimum Lee Distance

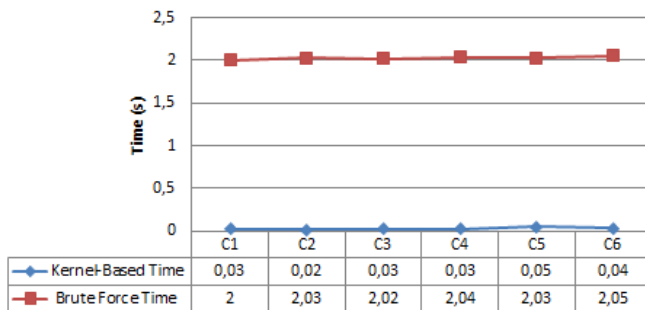


Figura 9. Temps de càlcul per la distància mínima de Lee per codis Z_2Z_4 .

Minimum Word

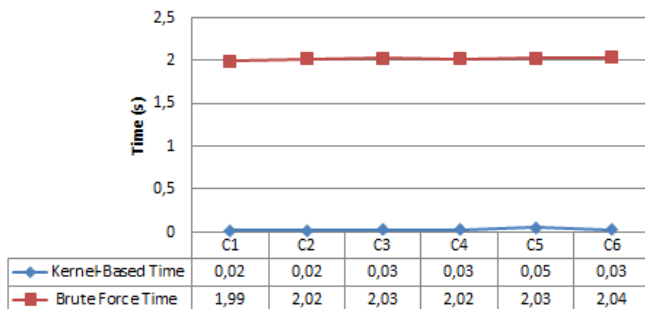


Figura 10. Temps de càlcul per la paraula-codi de pes mínim per codis Z_2Z_4 .

Minimum Words

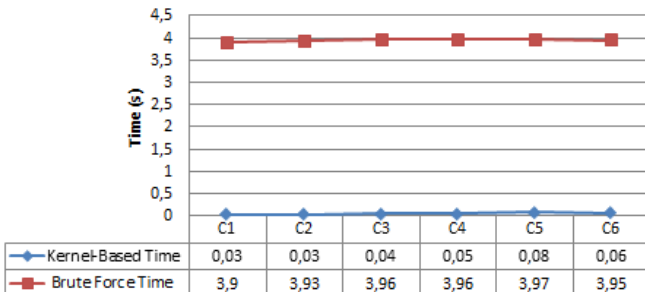


Figura 11. Temps de càlcul per les paraules-codi de pes mínim per codis Z_2Z_4 .

Weight Distribution

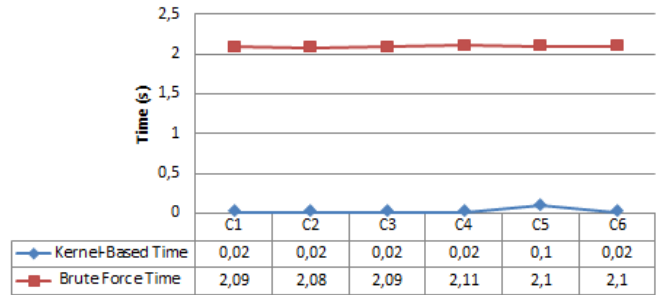


Figura 12. Temps de càlcul de la distribució de pesos per codis Z_2Z_4 .

Kernel Time

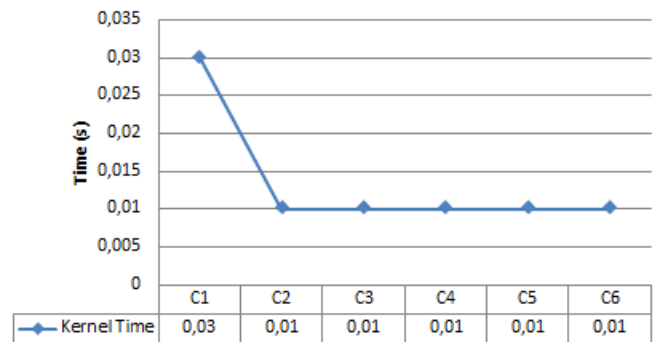


Figura 13. Temps de càlcul de la distribució de pesos per codis Z_2Z_4 .

En aquest conjunt de codis, l'algorisme basat en el càlcul del kernel i el basat en la força bruta impliquen uns temps de càlcul que difereixen de 2 a 3 segons, entre ells.

Tot i que els temps de còmput no estan al voltant de l'ordre de la vintena, com el cas anterior, l'algorisme basat en el càlcul del kernel emplea uns temps notablement més baixos per la computació de les mateixes funcions. En tots els casos, els temps emprats per les funcions són inferiors a 1s.

Els temps de càlcul d'aquesta família de codis presenta certa linealitat per cadascun dels algorismes estudiats sense que hi hagin canvis bruscos a mesura que augmenta el nombre de cosets representatius.

6 DISCUSSIÓ

En la primera etapa del projecte, mentre es programaven les funcions amb l'algorisme de la força bruta, el que es pensava a nivell teòric era que aquest tipus d'algorisme era poc eficient per realitzar les operacions ja que sempre calia recórrer totes les paraules-codi per obtenir la informació final desitjada. Aquesta forma no era òptima per estalviar temps de càlcul.

Posteriorment, quan es van implementar les funcions amb l'algorisme basat amb el càlcul del kernel, s'esperava que aquest fos més eficient ja que el kernel i els cosets d'un codi Z_2Z_4 -lineal es traslladen a l'anell d'enters mòdul 2 i, en aquest anell, s'usen les funcions binàries que ja estan optimitzades.

Dels resultats de la família de codis Hadamard, es detecta que l'algorisme òptim és el basat en la força bruta, justament el contrari del que es creia en un principi. S'observa que els codis amb temps de còmput més elevats són aquells en els quals el paràmetre m és igual a 11 i on la dimensió del kernel és la més gran.

Els codis de Hadamard contenen un codi dual (aquell generat per la matriu de comprovació H) relativament gran. Aquest fet implica que els algorismes, com el de la força bruta, que considerien només els vectors d'ordre quatre de la matriu generadora, G , tardin menys que no pas els que usin el dual en el seu còmput. La limitació observada resideix en la funció que calcula el kernel, tal com s'observa en el gràfic de còmput del kernel (figures 7) ja que aquests temps són substancialment elevats. Aquest fet implica que el càlcul del kernel és un coll d'ampolla a l'hora d'obtenir un rendiment més elevat dels temps de còmput en aquesta família de codis.

Actualment, pels codis de Hadamard és millor l'algorisme basat en la força bruta. No obstant això, si s'optimitzés la funció que calcula el kernel, és possible que els temps de les funcions implementades en aquest projecte milloressin i que l'algorisme basat en el càlcul del kernel desbanqués al de la força bruta.

Amb la segona família de codis, Codis Z_2Z_4 , sí que s'observa que l'algorisme basat en el càlcul del kernel és més eficient i òptim perquè va més ràpid en obtenir els resultats que no pas l'altre algorisme. En aquest cas, es destaca la gran diferència de temps a l'hora de dur a terme el càlcul del kernel. El temps invertit en aquest càlcul és de l'ordre de 0,01s mentre que en els codis de Hadamard pot arribar fins a 23s, tal com s'observa a les figures 7 i 13.

Com a conseqüència del càlcul del kernel i els cosets d'un codi a l'anell d'enters mòdul 2, es creia que l'augment del nombre de cosets estava directament relacionat amb el temps de còmput d'aquest algorisme. No obstant això, amb els resultats observats en les gràfiques anteriors dels Codis Z_2Z_4 no es produeix tal relació directa. El codi C_6 té el major nombre de cosets representatius, però en canvi, el codi que fa servir uns temps majors és el C_5 . Vists aquests resultats es pot concloure que el temps de còmput està vinculat amb la relació existent entre la dimensió del kernel i el nombre de cosets representatius.

A les dues famílies de codis emprades, s'observa que les funcions del càlcul del pes mínim de Lee i de la distància mínima de Lee obtenen els mateixos resultats. Aquest fet està causat per la programació d'aquestes funcions ja que és exactament la mateixa; és a dir, comparteixen el mateix codi font.

7 CONCLUSIONS

Durant la realització d'aquest projecte s'han assolit els objectius inicialment marcats que han estat coincidents amb els establerts durant el transcurs del mateix.

No s'ha pogut determinar amb certesa quins dels dos algorismes emprats és més eficient i òptim pel càlcul de les funcions ja que les proves realitzades amb les dues famílies de codis proporcionaven resultats dispars, però amb els indicis explicats en l'apartat anterior tot apuntaria que l'algorisme basat en el càlcul del kernel és més eficient. Tot i això, s'hauria d'optimitzar la funció usada pel càlcul del kernel i comprovar després si existeix una millora real del temps de còmput i si aquesta, optimitza els temps de les funcions desenvolupades en aquest projecte per l'algorisme basat en el kernel.

Existeix una altra possible futura línia d'investigació basada en la recerca de característiques de codis Z_2Z_4 -lineals amb la finalitat de descobrir per quins codis seria millor dur a terme les funcions usant l'algorisme basat en la força bruta i per quins codis és més eficient utilitzar les funcions fonamentades en l'algorisme basat en el càlcul del kernel. Així es podria obtenir el màxim rendiment de les dos tipus d'algorismes utilitzats a l'hora d'implementar totes les funcions.

AGRAÏMENTS

Especialment, a la Cristina Fernández per acceptar ser la tutora d'aquest projecte i, al mateix temps, per l'ajuda, les correccions i el suport proporcionat durant tot el procés. A la Mercè Villanueva per la l'ajuda facilitada.

A la meua família per donar-me l'oportunitat de poder estudiar i suportar-me durant els moments de molta pressió.

A l'Anna, l'Esther, el Carlos, el David, el Nil, l'Arcadi i el Lucas pels anys compartits, per creure en mi i per ser la distracció oportuna en moments molt necessaris.

BIBLIOGRAFIA

- [1] J. Borges, C. Fernández-Córdoba, J. Pujol, J. Rifà and M. Villanueva, "Z₂Z₄-linear codes: generator matrices and duality," *Designs, Codes and Cryptography*, vol. 54, pp. 167-179, 2010.
- [2] J. Borges, C. Fernández, J. Pujol, J. Rifà and M. Villanueva, "Z₂Z₄-linear codes. A Magma package," *Universitat Autònoma de Barcelona*, 2007.
- [3] J. Borges, C. Fernández, B. Gastón, J. Pujol, J. Rifà and M. Villanueva, "Z₂Z₄-Additive Codes. A MAGMA Package", *Universitat Autònoma de Barcelona*, 2009.
- [4] C. Fernández-Córdoba, J. Pujol, and M. Villanueva, "Z₂Z₄-linear codes: rank and kernel," *Designs, Codes and Cryptography* (July 2010) vol. 56. no. 1, pp. 43-59, ISSN: 0925-1022. DOI: 10.1109/TIT.2011.2119465.
- [5] B. Gastón, "Codis Z₂Z₄-Additius en MAGMA", *Projecte de final de carrera, Universitat Autònoma de Barcelona*, 2008.
- [6] M. Pujol, "Computing the Minimum Hamming Distance for Z₂Z₄-linear codes", *Projecte de final de màster, Universitat Autònoma de Barcelona*, 2012.
- [7] CCSG Development Group, "CCSG Style Guide", *Universitat*

Autònoma de Barcelona, 2011.

- [8] CCSG Development Group, “Breu Introducció al MAGMA”, Univeristat Autònoma de Barcelona, 2004.
- [9] Computational Algebra Group, “Overview of MAGMA V2.19 Features”, University of Sydney, 2016, <https://magma.maths.usyd.edu.au/magma/overview/pdf/overview219.pdf>.
- [10] “Handbook MAGMA”, <https://magma.maths.usyd.edu.au/magma/handbook/>

APÈNDIX

A1. SECCIÓ D'APÈNDIX

Les matrius usades en MAGMA pel càlcul del rendiment dels diferents algorismes, segons les funcions realitzades, són les següents:

```
M1 := Matrix(Z4,[[2,0,2,0,2, 0,0,0,0, 0,0,0,0,0,0 ],
[0,2,0,2,0, 0,0,0,0, 0,0,0,0,0,0 ],
[0,0,2,0,2, 0,0,2,0, 0,0,0,0,0,0 ],
[0,0,0,0,2, 0,0,0,2, 0,0,0,0,0,0 ],
[0,0,2,0,2, 0,0,0,0, 1,0,0,0,0,0 ],
[0,0,2,2,0, 0,0,0,0, 0,1,0,0,0,0 ],
[0,0,2,0,2, 0,0,0,0, 0,0,1,0,0,0 ],
[0,0,2,2,0, 0,0,0,0, 0,0,0,1,0,0 ],
[0,0,0,2,2, 0,0,0,0, 0,0,0,0,1,0 ],
[0,0,2,0,2, 0,0,0,0, 0,0,0,0,0,1 ]]);
```

```
C1 := Z2Z4AdditiveCode(M : Alpha := 5);
```

```
M2 := Matrix(Z4,[[2,0,2,0,2, 0,0,0,0, 0,0,0,0,0,0 ],
[0,2,0,2,0, 0,0,0,0, 0,0,0,0,0,0 ],
[0,0,2,0,2, 0,0,2,0, 0,0,0,0,0,0 ],
[0,0,0,0,2, 0,0,0,2, 0,0,0,0,0,0 ],
[0,0,2,0,2, 1,0,0,0, 1,0,0,0,0,0 ],
[0,0,2,2,0, 1,0,0,0, 0,1,0,0,0,0 ],
[0,0,2,0,2, 0,0,0,0, 0,0,1,0,0,0 ],
[0,0,2,2,0, 0,0,0,0, 0,0,0,1,0,0 ],
[0,0,0,2,2, 0,0,0,0, 0,0,0,0,1,0 ],
[0,0,2,0,2, 0,0,0,0, 0,0,0,0,0,1 ]]);
```

```
C2 := Z2Z4AdditiveCode(M : Alpha := 5);
```

```
M3 := Matrix(Z4,[[2,0,2,0,2, 0,0,0,0, 0,0,0,0,0,0 ],
[0,2,0,2,0, 0,0,0,0, 0,0,0,0,0,0 ],
[0,0,2,0,2, 0,0,2,0, 0,0,0,0,0,0 ],
[0,0,0,0,2, 0,0,0,2, 0,0,0,0,0,0 ],
[0,0,2,0,2, 1,0,0,0, 1,0,0,0,0,0 ],
[0,0,2,2,0, 1,1,0,0, 0,1,0,0,0,0 ],
[0,0,2,0,2, 0,1,0,0, 0,0,1,0,0,0 ],
[0,0,2,2,0, 0,0,0,0, 0,0,0,1,0,0 ],
[0,0,0,2,2, 0,0,0,0, 0,0,0,0,1,0 ],
[0,0,2,0,2, 0,0,0,0, 0,0,0,0,0,1 ]]);
```

```
C3 := Z2Z4AdditiveCode(M : Alpha := 5);
```

```
M4 := Matrix(Z4,[[2,0,2,0,2, 0,0,0,0, 0,0,0,0,0,0 ],
[0,2,0,2,0, 0,0,0,0, 0,0,0,0,0,0 ],
[0,0,2,0,2, 0,0,2,0, 0,0,0,0,0,0 ],
[0,0,0,0,2, 0,0,0,2, 0,0,0,0,0,0 ],
[0,0,2,0,2, 1,0,0,0, 1,0,0,0,0,0 ],
[0,0,2,2,0, 1,1,0,0, 0,1,0,0,0,0 ],
[0,0,2,0,2, 1,1,0,0, 0,0,1,0,0,0 ],
[0,0,2,2,0, 1,0,0,0, 0,0,0,1,0,0 ],
[0,0,0,2,2, 0,0,0,0, 0,0,0,0,1,0 ],
[0,0,2,0,2, 0,0,0,0, 0,0,0,0,0,1 ]]);
```

```
C4 := Z2Z4AdditiveCode(M : Alpha := 5);
```

```
M5 := Matrix(Z4,[[2,0,2,0,2, 0,0,0,0, 0,0,0,0,0,0 ],
[0,2,0,2,0, 0,0,0,0, 0,0,0,0,0,0 ],
[0,0,2,0,2, 0,0,2,0, 0,0,0,0,0,0 ],
[0,0,0,0,2, 0,0,0,2, 0,0,0,0,0,0 ],
[0,0,2,0,2, 1,0,0,0, 1,0,0,0,0,0 ],
[0,0,2,2,0, 1,1,0,0, 0,1,0,0,0,0 ],
[0,0,2,0,2, 1,1,0,0, 0,0,1,0,0,0 ],
[0,0,2,2,0, 1,1,0,0, 0,0,0,1,0,0 ],
[0,0,0,2,2, 0,1,0,0, 0,0,0,0,1,0 ],
[0,0,2,0,2, 0,0,0,0, 0,0,0,0,0,1 ]]);
```

```
C5 := Z2Z4AdditiveCode(M : Alpha := 5);
```

```
M6 := Matrix(Z4,[[2,0,2,0,2, 0,0,0,0, 0,0,0,0,0,0 ],
[0,2,0,2,0, 0,0,0,0, 0,0,0,0,0,0 ],
[0,0,2,0,2, 0,0,2,0, 0,0,0,0,0,0 ],
[0,0,0,0,2, 0,0,0,2, 0,0,0,0,0,0 ],
[0,0,2,0,2, 1,0,0,0, 1,0,0,0,0,0 ],
[0,0,2,2,0, 1,1,0,0, 0,1,0,0,0,0 ],
[0,0,2,0,2, 1,1,0,0, 0,0,1,0,0,0 ],
[0,0,2,2,0, 1,1,0,0, 0,0,0,1,0,0 ],
[0,0,0,2,2, 0,1,0,0, 0,0,0,0,1,0 ],
[0,0,2,0,2, 0,0,0,0, 0,0,0,0,0,1 ]]);
```

```
C6 := Z2Z4AdditiveCode(M : Alpha := 5);
```

```
M7 := Matrix(Z4,[[2,0,2,0,2, 0,0,0,0, 0,0,0,0,0,0 ],
[0,2,0,2,0, 0,0,0,0, 0,0,0,0,0,0 ],
[0,0,2,0,2, 0,0,2,0, 0,0,0,0,0,0 ],
[0,0,0,0,2, 0,0,0,2, 0,0,0,0,0,0 ],
[0,0,2,0,2, 1,0,0,0, 1,0,0,0,0,0 ],
[0,0,2,2,0, 1,1,0,0, 0,1,0,0,0,0 ],
[0,0,2,0,2, 1,1,0,0, 0,0,1,0,0,0 ],
[0,0,2,2,0, 1,1,0,0, 0,0,0,1,0,0 ],
[0,0,0,2,2, 1,1,0,0, 0,0,0,0,1,0 ],
[0,0,2,0,2, 1,0,0,0, 0,0,0,0,0,1 ]]);
```

```
C7 := Z2Z4AdditiveCode(M : Alpha := 5);
```