Codis binaris no lineals òptims: Propietats i construccions

Victòria González Benítez

Resum— Aquesta recerca té com a objectiu principal estudiar propietats i construccions dels codis binaris no lineals òptims i està dividida en tres parts. En primer lloc, estudiar els codis binaris no lineals 1-perfectes i trobar la relació entre el fet de ser sistemàtics i la dimensió del *kernel*. El *kernel* d'un codi és un subcodi lineal que permet mesurar la linearitat del codi. S'ha trobat que els codis amb dimensió del *kernel* més gran són sistemàtics. En segon lloc, estudiar els codis binaris òptims i construir-ne de coneguts per tal d'ampliar la base de dades que ja disposa el departament. Dels codis construïts, només el codi de Romanov té *kernel* màxim. També s'ha vist que els únics codis sistemàtics són el de Nadler code, el seu estès, l'Ostergard code i el seu estès. Finalment, l'últim objectiu ha consistit en intentar trobar nous codis binaris òptims no lineals generats a partir de codis binaris òptims o alguns dels millors codis coneguts i les construccions clàssiques: *directsum*, *extended*, *shortened* i *punctured code*. No s'ha trobat cap codi millor que els coneguts i s'ha arribat a la conclusió que l'optimalitat dels codis no es conserva quan s'apliquen les diferents construccions clàssiques.

Paraules clau—Codis binaris, codis no lineals, codis òptims, codis sistemàtics

Abstract—This research project has the main objective of studying the properties and the constructions of binary nonlinear optimal codes and it is divided in three parts. Firstly, study the binary nonlinear 1-perfect codes and find the relation between the fact of being systematics and the dimension of the *kernel*. The *kernel* of a code is a linear subcode that allows to measure the linearity of the code. It has been found that the codes with bigger dimension of the *kernel* are systematics. Secondly, study the binary optimal codes and build some of the known codes to extend the database of the department. From the constructed codes, only the Romanov code has a maximum *kernel*. Also it has been found that the unique systematic codes are the Nadler code, its extended code, the Ostergard code and its extended code. Finally, the last objective consisted on finding new binary nonlinear optimal codes generated from the known binary optimal codes or best codes and it has concluded that the optimality of the codes is not preserved when the different classical constructions are applied.

Index Terms—Binary codes, nonlinear codes, optimal codes, systematics codes

1 Introducció

PER realitzar una transmissió d'informació digital sobre un canal binari amb soroll es necessita transformar els missatges en seqüències de bits mitjançant un codi amb capacitat correctora d'errors. Tots els vectors binaris de longitud n possibles defineixen l'espai Z_2^n. Un codi binari C(n,M,d) de longitud n és un subconjunt de Z_2^n. Els elements de C s'anomenen paraules codi. La mida, M, del codi és el nombre de paraules codi que conté. La distància (de Hamming) entre dos vectors de longitud n ve donada pel nombre de bits que s'han de canviar d'un per aconseguir l'altre. De forma equivalent, si $x, y \in Z_2^n$ aleshores,

$$d_H(x,y) = \#\{\, i \mid x_i \neq y_i \,; 1 \leq i \leq n \,\}. \qquad (1)$$

La distància mínima, d, d'un codi és defineix com la menor de les distàncies entre dues paraules codi diferents qualssevol. La capacitat correctora, t, d'un codi és el nombre de bits erronis que es poden corregir per a cada paraula codi enviada en una transmissió i es calcula mitjançant la fórmula:

$$t = \lfloor (d-1)/2 \rfloor \qquad (2)$$

[13].

Un codi binari C(n,2^k,d) és lineal si C és un subespai de dimensió k de Z_2^n. En aquest cas, també es diu que és un codi C[n,k,d]. Aquests codis presenten bones propietats com l'eficiència en la codificació i l'existència de bons algoritmes de descodificació. A més, un codi lineal C pot ser representat amb una matriu generadora G, o sigui amb un conjunt més petit de paraules codi. La matriu generadora d'un codi lineal C[n, k,d] és qualsevol matriu G de mida k x n on les files formen una base de C. En general, hi ha diferents matrius generadores per a cada codi. Quan la matriu generadora té la següent forma:

$$G = (I_k \mid A), \qquad (3)$$

on I_k és una matriu identitat de mida k x k i A és una matriu de mida k x (n - k), aleshores direm que G està en forma estàndard [13].

- *E-mail de contacte: vickygonben@gmail.com*

- *Menció realitzada: Tecnologies de la Informació*
- *Treball tutoritzat per: Mercè Villanueva i Jaume Pujol (Department d'Enginyeria de la Informació i de les Comunicacions)*
- *Curs 2015/2016*

Un codi binari C(n,2^k,d) és sistemàtic si existeix un conjunt de coordenades $(i_1, i_2, \dots, i_k)$, normalment les primeres, tal que restringint totes les paraules codi en aquestes posicions obtenim un conjunt amb la mateixa mida que el codi C. Per a un codi sistemàtic, les k coordenades $(i_1, i_2, \dots, i_k)$ corresponen als bits d'informació (*information set*), mentre que la resta de les $(n - k)$ coordenades corresponen als bits de comprovació (*parity check set*) [13]. En un codi lineal, cada conjunt de k columnes linealment independents d'una matriu generadora correspon a un conjunt de posicions que formen un conjunt d'informació de C [13].

Tot i que no presenten, com els lineals, tant bones propietats per codificar i descodificar, ens trobem que alguns dels millors codis que existeixen són no lineals. Si fixem una longitud n i una distància mínima d, direm que un codi C(n,M,d) és òptim si conté el nombre màxim de paraules codi M. En aquest treball considerem que tots els codis no lineals contenen el vector zero. Per als codis no lineals no existeix una matriu generadora G que permeti representar tot el codi, però es poden representar a partir d'un subconjunt lineal, anomenat nucli (*kernel*), i els representants dels traslladats d'aquest nucli. Donat un codi binari C(n,M,d) el *kernel* d'aquest codi es defineix com el conjunt de totes les paraules codi que deixen el codi invariant al fer translacions, és a dir,

$$\ker(C) = \{x \in C : x + C = C\} \qquad (4)$$

[17]. El *kernel* de C és un subcodi lineal de C. En general, C pot ser escrit com la unió de traslladats del *kernel* de C, és a dir $C = \bigcup_{i=0}^{t} (\ker(C) + c_i)$. Aquestes translacions $\ker(C) + c_i$ s'anomenen *cosets* o traslladats de C, i $\{c_0, c_1, c_2, \dots, c_t\}$ és un conjunt de representants d'aquests *cosets* [17].

Durant aquest projecte s'han calculat alguns paràmetres importants dels codis que es definirán a continuació. En primer lloc, per a dur a terme l'anàlisi de les dades resultants és important conèixer aspectes com el *rank* i *l'span*. El *rank* d'un codi binari C, r = rank(C), és la dimensió de *l'span* de C. *L'span* és el codi binari lineal generat per les paraules codi de C [19]. El radi de recobriment (*covering radius)* d'un codi C, denotat per $\rho(C)$, és l'enter més petit s tal que Z_2^n és la unió de totes les esferes de radi s que tenen com a centre les paraules codi de C, o equivalentment,

$$\rho(C) = max\{d_H(x, C) \mid x \in Z_2^n\}, \qquad (5)$$

on $d_H(x, C) = \min\{d_H(x, C) \mid c \in C\}$ [25]. En un codi perfecte es compleix que la capacitat correctora és igual al radi de recobriment, $\rho(C) = t$.

També es faran servir les construccions clàssiques per generar nous codis no lineals òptims. En primer lloc, a partir de dos codis C i D, el *direct sum* és el codi binari resultant que consisteix en tots els vectors de la forma (u,v), on $u \in C$ i $v \in D$. El e*xtended code* és el codi C' construït a partir d'afegir una coordenada extra a cada vector de C, de tal manera que la suma de totes les coordenades és zero. El *punctured code* de C en la coordenada número j, anomenat C^j, és el codi que consisteix en les paraules codi de C després d'eliminar la coordenada de la posició j de cada paraula codi de C. El *shortened code* de C en la coordenada número j, denotat per C_0^j, és el codi que consisteix en considerar les paraules de C que tenen un zero en la coordenada número j, i eliminar aquesta coordenada [25].

En aquest treball s'estudiaran principalment els codis binaris òptims. Entre aquests una de les clases o famílies més importants són els codis perfectes. Un codi binari de longitud n és 1-perfecte si qualsevol vector binari de Z_2^n està a distància 1 o 0 d'una paraula codi de C. Sempre tindran una longitud n = 2^m - 1 i distància mínima d = 3. Un exemple d'aquest tipus de codi són els codis de Hamming [13], [17].

Malgrat l'existència de diferent software lliure matemàtic com és SAGE [24], MAPLE [29] o MATLAB [28], per a la realització d'aquest projecte s'ha triat MAGMA. Principalment perquè les llibreries del Departament d'Enginyeria de la Informació i de les Comunicacions (DEIC) amb les que es treballarà estan programades en aquest llenguatge [19]. Actualment, MAGMA és un dels millors softwares dissenyat per resoldre problemes computacionals difícils en àlgebra, i sobretot, en teoria de codis [5], [19] [26], [27].

1.1 Estat de l'art

Aquest treball s'ha basat en estudiar els codis no lineals binaris perfectes i òptims, dues de les famílies més importants de codis. Alguns dels millors codis que existeixen són no lineals, però tot i així encara no es coneix molt sobre aquests codis. Un autor important en aquest camp és F. J. MacWilliams amb el seu treball *The theory of error-correcting codes* [13]. Aquest treball s'ha desenvolupat a partir del treball elaborat per Fanxuan Zeng en la seva tesi doctoral titulada: *Nonlinear codes: Representation, constructions, minimum distance computation and decoding* [25].

Existeixen algunes bases de dades o pàgines web on apareixen els paràmetres n, k, d, dels codis òptims o dels millors codis coneguts, són taules recopilades per autors com Litsyn o Brouwer. El problema és que aquestes fonts [2], [12] no contenen explícitament la construcció dels codis ni s'analit-za com són de no lineals aquests codis òptims a partir dels invariants *rank* i dimensió del *kernel*.

1.2 Objectius del treball

Aquest treball s'ha dividit en tres temes o objectius principals:

1. Estudiar els codis binaris 1-perfectes.
2. Estudiar i construir codis binaris òptims coneguts.
3. Construir nous codis binaris òptims.

En primer lloc, s'han estudiat els codis 1-perfectes binaris de longitud 15 i els seus estesos (de longitud 16). Al DEIC s'havia construït una base de dades amb tots aquests codis binaris de longitud 15 i 16 [7], [15], [19]. Concretament en aquest treball, s'ha estudiat si són, o no, sistemàtics; amb l'objectiu de trobar la relació entre la dimensió del *kernel* i si el codi és, o no, sistemàtic i, en cas que hagi resultat ser sistemàtic, s'ha calculat un conjunt d'informació.

En segon lloc, s'han estudiat codis binaris òptims. Existeixen taules amb informació sobre codis òptims on es poden trobar els codis agrupats segons la distància mínima i amb les referències per poder construir-los [2], [12]. Al DEIC existeix una base de dades amb alguns d'aquests codis ja construïts [19]. Per tant, la segona part del treball ha consistit en construir alguns dels codis òptims o dels millors codis coneguts (per ampliar la base de dades del departament) i s'ha estudiat la relació entre la dimensió del *kernel* d'un codi i la seva optimalitat. El procediment ha consistit en calcular: la longitud, la mida, la distància mínima, el *covering radius*, la dimensió del *kernel*, la distància mínima del *kernel*, el *covering radius* del *kernel*, la mida del *span*, la distància mínima del *span* i el *covering radius* del *span*; de cada codi i després del seu *extended* o *punctured code* (en funció de si la distància mínima del codi és parell o senar).

Per acabar, l'últim punt que s'ha tractat al treball és la construcció de nous codis a partir de les construccions clàssiques aplicades als codis òptims coneguts de la base de dades anomenada anteriorment. S'ha utilitzat una llibreria per a codis no lineals binaris en MAGMA [19] basada en els resultats de la tesis del Dr. Fanxuan Zeng [25]. Existeixen construccions que permeten construir nous codis a partir d'altres. Les que s'han utilitzat en aquest treball són: *extended code, punctured code, direct sum code i shortened code*. S'han estudiat les propietats mètriques d'aquests nous codis i s'han relacionat amb la dimensió del seu *kernel*.

La metodologia utilitzada en la realització d'aquest treball ha estat una combinació de la metodologia matemàtica, en la part teòrica del treball, i d'altra banda, metodologia experimental, en la part pràctica. S'explica amb més detall en el següent apartat.

Aquest document es divideix en els següents apartats: primer de tot es parla sobre la metodologia utilitzada, en segon lloc s'explica com s'ha dut a terme la investigació, en el següent apartat es mostren els resultats obtinguts a partir de la recerca, i finalment, es mostraran les conclusions extretes a partir de la realització d'aquest treball, els agraïments, les referències bibliogràfiques utilitzades i finalment, els apèndixs.

2 METODOLOGIA

Al tractar-se d'un tema de recerca sobre teoria de codis, gran part del treball s'ha basat en estudiar teoria matemàtica per poder encaminar les possibles solucions.

Per tant, per a la realització d'aquest treball s'ha plantejat una metodologia mixta: per una part, metodologia matemàtica per explorar el problema i les possibles solucions i d'altra banda, una metodologia experimental per a desenvolupar les solucions triades.

La metodologia matemàtica no té uns passos definits ja que depèn del problema al que ens afrontem. Actualment, es redacten més de 400 pàgines amb només els abstracts dels papers que es publiquen cada mes, la matemàtica es troba en una època daurada ja que es produeixen quantitat i qualitat de descobriments matemàtics [4]. Per tant, l'evolució de les solucions matemàtiques es desenvolupa a bon ritme.

Com diuen els principis de la metodologia matemàtica, en primer lloc, és clau tenir clar els projectes relacionats que s'han realitzat fins ara (*Blue sky research*), que és el que ha sortit bé i els errors que s'han produit. Per això, cal documentar-se bé per poder saber on estem i cap a on podem encaminar la solució. També és important trobar un mètode estàndard per a un tipus de problema estàndar, és a dir, el millor que es pot fer per començar és reduir el problema a un dels que ja en coneixem la solució; així que la comparació amb els codis lineals (dels que tenim molta més informació que dels no lineals) és un punt important per començar a conèixer les característiques bàsiques dels no lineals [4].

La metodologia experimental que es va utilitzar és el *Test Driven Development* (TDD). Aquesta metodologia consisteix en pensar els casos importants que ha de tractar el sistema per després codificar les solucions basades en aquests "casos d'ús". Per tant, ha estat convenient utilitzarla en la realització d'aquest treball ja que a partir de l'estudi teòric previ (via la metodologia matemàtica) es va decidir quins eren els casos importants a tenir en compte i després es van generar les funcions per arribar a la solució pensada per després poder evaluar els resultats [22], [23].

En primer lloc, es van definir les característiques i requisits del problema que es volia resoldre. També es van dissenyar i es van documentar les funcions que es necessitaven per elaborar els experiments.

En segon lloc, es van dissenyar els tests que es van fer servir per validar el desenvolupament de les funcions proposades. Després d'això es van dissenyar i es van implementar les funcions per resoldre el problema.

Es van executar els tests dissenyats i es van recollir els resultats. Si el test no era correcte, es van tornar a dissenyar i implementar les funcions. Finalment, es van interpretar els resultats obtinguts segons els objectius proposats.

3. DESENVOLUPAMENT DEL PROJECTE

Inicialment, es va fer un treball de cerca d'informació per poder estudiar les propietats bàsiques dels codis lineals i no lineals, i també conèixer el llenguatge de programació MAGMA i la llibreria amb la que es treballaria. Les primeres tasques van ser: consultar llibres sobre teoria de codis [13], assistir a classes de màster de la doctora Mercè Villanueva, consultar tesis i treballs anteriors relacionats amb el tema [8], [25], mirar tutorials i documentació de MAGMA [5], [26], [27], i finalment, consultar la documentació sobre la llibreria per a codis no lineals binaris del DEIC [19].

3.1. Codis binaris 1-perfectes

Per començar amb el primer objectiu del treball, es va realitzar un estudi sobre codis binaris 1-perfectes per determinar quins d'ells eren sistemàtics amb l'objectiu de trobar quina és la relació entre la dimensió del *kernel* i el fet de ser o no un codi sistemàtic 1-perfecte.

Per començar amb aquesta tasca, es va revisar la documentació de la base de dades de codis perfectes de MAGMA (amb codis de longitud 15 i els seus estesos de

longitud 16) per saber com accedir i treballar amb els diferents codis [19]. Després, es va crear una funció anomenada: `BinaryPerfectCodes15DB()` per escriure en diferents fitxers (un per a cada *rank* diferent) els següents paràmetres de cada codi binari perfecte de longitud 15 de la base de dades: el *rank*, la dimensió del *kernel*, el número (paràmetre que serveix per indexar els codis dins de la base de dades), si és sistemàtic, i en el cas de ser-ho, el seu conjunt d'informació. En [20] es construeixen per primera vegada codis 1-perfectes que no són sistemàtics.

Es coneix que hi ha 5983 codis binaris 1-perfectes no equivalents de longitud 15. La distribució d'aquests codis, segons el *rank* i la dimensió del *kernel* ve documentada en la Taula 1 [15], [19]:

TAULA 1
CODIS BINARIS 1-PERFECTES DE LONGITUD 15

	Dimension of the kernel											
Rank	1	2	3	4	5	6	7	8	9	10	11	Total
11											1	1
12							12	3	3			18
13				224	262	176	28	13				703
14		163	1287	2334	941	129	8	1				4863
15	19	14	8	338	19							398
												5983

Amb la informació extreta dels fitxers generats amb la funció `BinaryPerfectCodes15DB()`, s'ha construït la Taula 2, en la que s'indica el número de codis no sistemàtics en funció del *rank* i la dimensió del *kernel*.

TAULA 2
CODIS BINARIS 1-PERFECTES DE LONGITUD 15 NO SISTEMÀTICS

	Dimension of the Kernel		
Rank	1	2	Total
14		1	1
15	10	2	12
			13

Després d'això es va fer el mateix amb els codis binaris 1-perfectes estesos de longitud 16 de la base de dades, `BinaryPerfectCodes16DB()`.

També es coneix que existeixen 2165 codis binaris 1-perfectes estesos no equivalents de longitud 16. La distribució d'aquests codis, segons el *rank* i la dimensió del seu *kernel* ve documentada en la Taula 3 [15], [19]:

TAULA 3
CODIS BINARIS 1-PERFECTES ESTESOS DE LONGITUD 16

	Dimension of the kernel											
Rank	1	2	3	4	5	6	7	8	9	10	11	Total
11											1	1
12							8	2	2			12
13				82	89	67	11	7				256
14		102	449	786	326	53	4	1				1721
15	18	14	8	123	12							175
												2165

Amb la informació extreta dels fitxers generats per la funció `BinaryPerfectCodes16DB()`, s'ha construït la Taula 4 com a resum, amb informació sobre el nombre de codis que no són sistemàtics, indexada pel *rank* i la dimensió del *kernel*.

TAULA 4
CODIS BINARIS 1-PERFECTES ESTESOS DE LONGITUD 16 NO SISTEMÀTICS

	Dimension of the Kernel		
Rank	1	2	Total
14		1	1
15	9	2	11
			12

Com es pot observar, hi ha un codi menys que als de longitud 15, això vol dir que hi ha dos codis no sistemàtics de longitud 15 que, al fer el seu estès dóna el mateix codi de longitud 16. Per detectar aquests dos codis, el que s'ha fet ha estat fer el codi estès de cadascun dels 13 codis de longitud 15 i utilitzar la funció `BCDClassification(DB16,D);` per trobar a quin codi de longitud 16 correspon.

3.2 Codis binaris òptims coneguts

La segona part d'aquest treball va consistir en construir codis binaris òptims coneguts [3]. El que es va fer en primer lloc va ser crear un fitxer que generava un codi amb una distància mínima d. Si la distància era senar es va generar també el seu *extended code* i, si era parell, es va generar també el seu *punctured code*.

Per generar *l'extended code*, s'ha utilitzat la funció `BinaryExtendedCode(C)`, que funciona de la següent manera: passant-li un codi binari C genera un nou codi binari C' a partir d'afegir una coordenada extra a cada paraula codi de C de tal manera que la suma de les coordenades fos zero.

Per generar el *punctured code*, s'ha creat una nova funció anomenada `BinaryMaxPuncturedCode(C)`, que donat un codi binari C de longitud n, construeix un nou codi binari C'' a partir de cridar a la funció `BinaryPunctureCode(C,i)` n vegades, retornant només el millor codi que es construeix, és a dir, el que té la distancia mínima major. La funció `BinaryPunctureCode(C,i)`, s'ha encarregat d'anar eliminant una certa columna i ($1 \leq i \leq n$) de cada paraula codi de C.

Es va començar construint codis de distància mínima 3 i 4: primer amb els codis de repetició, seguint amb els codis lineals de MAGMA, els codis de Hamming, i per acabar amb els codis no lineals com per exemple: Julin code, generat a partir del conjunt de paraules codi (mida 72 i 144) [10]; el Best code, generat a partir de quatre vectors generadors als quals se'ls van fent rotacions i afegint al conjunt de paraules codi del Best code [1]; el Romanov code [21]; o el Hamalainen code [9]. Es va continuar amb els codis de distància mínima 5 i 6 no lineals com són: Nadler code, que es construeix a partir del conjunt sencer de paraules codi

[14]; o el Plotkin code, que es construeix també a partir del conjunt de totes les paraules codi [13], [18]. I finalment, amb els codis de distància mínima 9 i 10 no lineals: Kaikkonen code [11]; els Ostergard codes [16]; i per acabar, el Elssel-Zimmermann code [6].

A partir d'aquestes construccions es van calcular les següents propietats dels diferents codis: si eren sistemàtics (en cas afirmatiu, també el seu conjunt d'informació), longitud, el cardinal del codi, distància mínima, radi de recobriment, cardinal del *kernel*, distància mínima del *kernel,* radi de recobriment del *kernel,* cardinal de l'*span,* distància mínima de l'*span i* radi de recobriment de l'*span*.

Per dur a terme aquesta tasca, es va crear un fitxer en MAGMA amb un nom del tipus: `BBCnxdxMx_gen.m`, on n correspon a la longitud del codi, d a la distància mínima i M a la mida del codi; per a que al generar els codis òptims, o millors, coneguts també s'anessin calculant els paràmetres anomenats anteriorment mitjançant la crida a diferents funcions del paquet Binary Codes [19]. Aquest fitxer acabat en _gen, genera dos fitxers de resultats: un amb el codi òptim que es vol contruir i el segon fitxer amb el seu *extended code* o *punctured code* (segons si la distància mínima és senar o parell).

També es va calcular si d'aquest conjunt de codis, els no lineals tenien *kernel* màxim o no, és a dir, si la dimensió del *kernel* era el més gran possible. Això es calcula de la següent manera: si el nombre d'elements del codi és una potencia de dos, el nombre de *cosets* ha de ser 4; si la mida del codi no és una potència de dos, el nombre de *cosets* ha de ser senar. El nombre de *cosets* es calcula a partir de fer la divisió entre el nombre de paraules codi del codi i el nombre de paraules codi del *kernel*.

3.3 Construcció de nous codis binaris òptims

Finalment, l'última part del treball va consistir en, fent servir els codis òptims ja construïts i les construccions clàssiques conegudes: *direct sum code, extended code, punctured code, i shortened code*; construïr nous codis (amb intenció de que fossin nous codis òptims o millors), estudiar les seves propietats mètriques i analitzar si l'optimalitat dels codis es conserva si apliquem les construccions clàssiques.

Per fer les construccions de nous codis el criteri que es va triar va ser el següent: en primer lloc aplicar *directsum* a una parella de codis òptims (dels construïts anteriorment) i sobre el codi resultant generar tres codis, el *extended*, el *shortened* i el *punctured code*.

Per aconseguir-ho, s'ha dissenyat una funció anomenada `BinaryCodeDirectSumConstruction()` que, a partir del conjunt de codis òptims o millors coneguts, aplica el criteri de construcció anomenat anteriorment sobre cada parella de codis possibles, és a dir, cada codi amb tots els demés codis del conjunt (incloent ell mateix).

A l'hora d'aplicar el *shortened code* s'hauria d'haver creat una funció que triés d'entre els n codis el millor, per reduir el nombre de codis per analitzar, ja que el que s'ha fet ha estat fer aquesta construcció n vegades sobre el codi de longitud n; això ha fet que el fitxer resultant de dades fos molt gran i ha dificultat la lectura i l'anàlisi de les dades. Aleshores, el que s'ha fet de manera manual va ser, per cada conjunt de codis generat amb la funció `BinaryShortenedCode()`, triar el que tenia major distància mínima.

La funció `BinaryCodeDirectSumConstruction()` també s'ha encarregat de calcular la longitud, la distància mínima, la mida, la dimensió del *kernel* i el *rank* i els resultats es van escriure en un fitxer (tenim un fitxer diferent per a cada tipus de construcció). Aquesta informació es va recopilar en taules amb l'objectiu d'analitzar els paràmetres i trobar nous codis òptims o millors [2], [12].

Durant la durada del projecte, es va treballar amb totes les parelles de codis òptims coneguts de la base de dades del departament, fetes a partir dels 13 primers codis combinats amb tots els demés (46 codis en total). Les dades es van passar a quatre taules resum que contenien aproximadament uns 2.000 codis (molts d'ells equivalents), així que per analitzar les dades resultants primer es va fer una criba amb l'intenció de triar el millor codi de cada combinació possible (n,d), és a dir, el que tingués major nombre de paraules (M). Després d'aquesta criba, van quedar 130 codis a analitzar.

Finalment, es va comparar cadascun dels codis amb les taules de codis òptims o millors [2], [12] per veure si alguns d'ells eren òptims.

Per realitzar aquesta comparació (veure si la M dels nous codis construïts era major que la dels millors codis coneguts) es van haver de tenir en compte diferents aspectes. En primer lloc, es van utilitzar les taules de Litsyn [12], ja que són les que estan més actualitzades i donen fites inferiors de M per a tots els casos. En segon lloc, aquestes taules venen indexades per n, d i donen com a paràmetres q i N. A partir d'aquests paràmetres s'ha calculat la M de cada codi utilitzant la fórmula:

$$M = N \cdot 2^q. \tag{6}$$

En aquestes taules només apareixen els codis amb d des de 3 fins a 29 senars, ja que es pot calcular la M per als codis amb d parell utilitzant l'equivalència:

$$M(n,d) = M(n-1,d-1). \tag{7}$$

M(n,d) representa el cardinal del codi que té una longitud n i com a distància mínima d. Per tant, si es calcula la M per a un codi de la taula amb d senar, directament se sap que el codi amb d + 1 i n + 1 tindrà la mateixa fita inferior per a M. Per últim, en aquestes taules es pot trobar la fita inferior de M per als codis que tenen una n <= 512. Hi ha alguns dels valors de n en aquest rang que no apareixen a la taula (amb d senar), per tant s'ha de calcular. En primer lloc s'han de mirar els paràmetres k i N del següent valor de n més proper (sempre ha de ser major) a la n que es vol trobar (denotada en la fórmula com a n'). Aleshores se sap que:

$$M(n',d') \geq N \cdot 2^{(k-(n-n'))}. \tag{8}$$

4 Anàlisi de resultats

Com a primer resultat de la primera part del treball, s'ha trobat que hi ha tretze codis 1-perfectes no sistemàtics de

longitud 15 mentre que de longitud 16, és a dir, dels seus estesos, n'hi ha dotze. Com s'ha comentat en l'apartat anterior, es va fer una classificació per veure quins codis de longitud 15 compartien codi estès. En la Taula 5 es poden veure els codis perfectes no sistemàtics de longitud 15 i en l'altra els de longitud 16. Estàn marcats en negreta els dos codis de longitud 15 que donen com a resultat el mateix codi estès, també marcat en negreta a la taula dels codis de longitud 16.

TAULA 5
CODIS BINARIS 1-PERFECTES NO SISTEMÀTICS

Nonsystematic perfect codes [Rank, KerDim, num]
[14,2,163], [15,1,1], **[15,1,6], [15,1,7]**, [15,1,11], [15,1,12], [15,1,13], [15,1,15], [15,1,16], [15,1,17], [15,1,18], [15,2,9], [15,2,10]
Nonsystematic extended perfect codes [Rank, KerDim, num]
[14,2,102], [15,1,1], **[15,1,6]**, [15,1,10], [15,1,11], [15,1,12], [15,1,14], [15,1,15], [15,1,16], [15,1,17], [15,2,9], [15,2,10]

A partir de l'estudi dels codis 1-perfectes de longitud 15 i 16, s'ha obtingut com a conclusió que tots els codis amb *kernel* "gran", és a dir, codis amb una part lineal més gran, són sistemàtics.

Els resultats de la segona part del treball, és a dir, les dades obtingudes dels codis òptims o millors coneguts de la base de dades del DEIC i els construïts coneguts, es poden veure en la taula resum de l'Apèndix A.

Com ja se sap, tots els codis lineals són sistemàtics (siguin òptims o no), però això no és així en el cas dels codis no lineals. Si tenim un codi lineal sistemàtic, és fàcil trobar un codi equivalent que, en conseqüencia, també serà sistemàtic. En el cas de tenir un codi no lineal no sistemàtic, es podria trobar un codi no equivalent però amb els mateixos paràmetres que si fos sistemàtic. Encara que aquesta no és una tasca senzilla. En la taula de l'Apèndix A, estan marcats en negreta els codis no lineals òptims que són sistemàtics.

Com a primera conclusió d'aquesta part es pot dir que els codis no lineals òptims no són sempre sistemàtics. S'ha trobat que el codi de Nadler (i el seu estès) i el d'Ostergard (i el seu estès) són codis no lineals òptims sistemàtics.

En segon lloc, dels codis no lineals construïts, n'hi ha amb diferents dimensions de *kernel*, per tant no podem dir que l'optimalitat dels codis no lineals depèn de la mida del seu nucli. Al calcular si els codis no lineals òptims coneguts tenen el *kernel* màxim, s'ha trobat que del conjunt de codis construïts, només el codi de Romanov té *kernel* màxim.

Finalment, els resultats de la tercera part són els següents. A partir de la construcció de nous codis òptims o millors s'han obtingut quatre fitxers de MAGMA. Aquesta informació s'ha passat a taules excel per reduir-ne el volum. Finalment, s'han analitzat per parelles (n, d) i s'han triat els millors codis, 130 codis. Amb aquesta informació s'ha construït una taula, que es pot veure a l'Apèndix B, amb els paràmetres dels nous codis i quins són els passos per construir-los.

Com s'ha explicat en l'apartat anterior, aquests codis s'han comparat amb els codis òptims coneguts per trobar si algun dels nous construïts era òptim o dels millors codis. Després de fer tota la comparació de dades no s'ha trobat cap codi millor que els que ja es coneixen.

5. CONCLUSIONS

Aquest treball s'ha dividit en tres parts. El primer objectiu ha estat trobar la relació entre la dimensió del *kernel* i el fet de que un codi binari no lineal perfecte (en el cas d'aquest treball, s'ha treballat amb codis 1-perfectes de longitud 15 i 16) fos o no sistemàtic. Com ja se sap, els codis lineals sempre són sistemàtics, però això no és així en el cas dels codis no lineals. A partir de la realització d'aquest treball s'ha pogut veure que en els codis no lineals 1-perfectes de longitud 15 i els seus estesos com més gran és la part lineal del codi, és a dir, la dimensió del *kernel* és més gran, més possibilitats hi ha de que aquest codi sigui sistemàtic. Un altre aspecte que s'ha vist és que, com que hi ha menys codis en el conjunt dels estesos (longitud 16), si fem l'estès d'alguns dels codis binaris no lineals 1-perfectes de longitud 15 ens trobem que hi ha solapaments dels seus estesos. Això s'ha pogut demostrar al fer un anàlisi dels codis de longitud 15 que no eren sistemàtics utilitzant la funció `BCDClassification()` per classificar els seus estesos.

En aquest treball s'han estudiat els codis de longitud 15 i 16 ja que es tenien tots a la base de dades del DEIC. Una possible extensió d'aquest treball per dur-la a terme en un futur podria ser estudiar els codis 1-perfectes de longitud 31 i 32, que s'obtenen a partir de construccions amb codis de longitud 15 i 16. Un objectiu seria trobar si a partir de construccions amb codis: no sistemàtic + no sistemàtic, es crea un codi també no sistemàtic, és a dir, si es conserva la propietat de no ser sistemàtic.

El segon objectiu d'aquest treball ha estat estudiar la mida del kernel per alguns codis binaris òptims, i alguns dels millors codis coneguts. En relació a aquest aspecte, el primer que s'ha aconseguit amb aquest treball és afegir alguns codis binaris òptims a la base de dades del DEIC. En segon lloc, no s'ha pogut confirmar la hipòtesi de que l'optimalitat d'un codi no lineal depèn de la mida del seu *kernel*, ja que hi havia codis amb *kernels* de mides molt diferents. Per tant, com a segona conclusió es va veure que cap dels codis no lineals d'aquest conjunt, excepte el codi de Romanov, tenia *kernel* màxim. Sobre els codis d'aquesta base de dades també es va calcular si eren sistemàtics o no. Tampoc es va poder extreure una conclusió sobre el fet de ser sistemàtic i la dimensió del *kernel*, ja que els únics codis que són sistemàtics són: el codi de Nadler (i el seu estès) i el d'Ostergard (i el seu estès).

L'objectiu final d'aquest treball ha estat construir nous codis binaris òptims a partir dels millors codis coneguts construïts en la segona part del treball i les construccions clàssiques: *directsum*, *extended*, *shortened* i *punctured code.* Després de la recerca, no s'ha trobat cap codi òptim. Per manca de temps no s'han construït tots els codis que s'havien pensat construir, per tant en un futur treball es podria acabar d'aplicar aquest criteri de construcció (en primer lloc aplicar *directsum* a una parella de codis òptims, construïts anteriorment, i sobre el codi resultant generar tres codis, el *extended*, el *shortened* i el *punctured code*) amb els codis restants. Una altra possible extensió d'aquesta part del treball podria ser canviar el criteri de construcció.

Com a principal conclusió d'aquesta última part s'ha extret que l'optimalitat dels codis no es conserva al aplicar les construccions clàssiques.

AGRAÏMENTS

L'autora vol agraïr el gran recolzament que ha rebut dels seus tutors, Mercè Villanueva i Jaume Pujol. Ha estat un plaer treballar amb vosaltres.

BIBLIOGRAFIA

[1] M. R. Best, "Binary Codes with a Minimum Distance of Four," IEEE Transactions on Information Theory, vol. 26, no. 6, November 1980.

[2] A. E. Brouwer, "Table of general binary codes", http://www.win.tue.nl/~aeb/codes/binary-1.html

[3] A. E. Brouwer, J. B. Shearer, N. J. A. Sloane and W. D. Smith, "A new Table of Constant Weight Codes," IEEE Transactions on Information Theory, vol. 36, no. 6, November 1990.

[4] R. Brown and T. Porter, The methodology of mathematics, Dept. de geometría e topología at the University of Sevilla, November 1993.

[5] J. J. Cannon and C. Playoust, "First Steps in MAGMA," School of Mathematics and Statistics, Australia, August 1996; http://magma.maths.usyd.edu.au/magma/pdf/first.pdf

[6] K. Elssel, K. H. Zimmermann, "Two New Nonlinear Binary Codes," IEEE Transactions on Information Theory, vol 51, no. 3. pp. 1189-1190, 2005.

[7] T. Etzion and A. Vardy, "Perfect Binary Codes: Constructions, Properties, and enumeration," IEEE Transactions of Information Theory, vol. 40, no. 3, May 1994.

[8] M. Gutiérrez, "Codis no lineals en MAGMA: Noves Construccions," degree final project, Escola d'Enginyeria UAB, Juny 2010.

[9] H. O. Hamalainen, "Two New Binary Codes with Minimum Distance Three," IEEE Transactions on Information Theory, vol 34, no. 4. pp. 875 - 876, 1988.

[10] D. Julian, "Two Improved Block Codes," Correspondance, pp. 439-440, 1965.

[11] M.K. Kaikkonen, "A new four-error-correcting codes of length 20." IEEE Transactions of Information Theory, vol 35, no. 6. pp. 1344 - 1345, 1989.

[12] S. Litsyn, E. M. Rains and N. J. A. Sloane, "Table of NonLinear binary codes.," November 24, 1999; http://www.eng.tau.ac.il/~litsyn/tableand

[13] F. J. MacWilliams, *The theory of error-correcting codes*., North-Holland Mathematical Library, 1977.

[14] M. Nadler, "A 32-Point n = 12, d = 5 Code*," IEEE Transactions on Information Theory, vol. IT-6, pp. 445-450, April 1961.

[15] P. R. J. Östergård and O. Pottonen, "The Perfect Binary One-Error-Correcting Codes of Length 15: Part I-Classification," IEEE Transactions of Information Theory, vol. 55, no. 10, October 2009.

[16] P. R. J. Ostergard, "Two New Four-Error-Correcting Binary Codes," Designs, Codes and Cryptog., vol 36, pp. 327-329, 2005.

[17] K. T. Phelps and M. Levan, "Kernels of Nonlinear Hamming Codes," Designs, Codes and Cryptography, 6 247-257, 1995.

[18] M. Plotkin, "Binary Codes with Specified Minimum Distance," IRE Transactions on Information Theory, pp. 445-450, September 1960.

[19] J. Pujol and M. Villanueva, "Binary codes: binary codes databases, MAGMA Packages," Combinatorics, Coding and Security Group (CCSG), Barcelona, December 17, 2014.

[20] A. M. Romanov, "On nonsystematic perfect binary codes of length 15," Discrete Applied Mathematics, vol. 135, pp. 255-258, 2004.

[21] A. M. Romanov, "New Binary Codes with Minimum Distance 3," Probl. Peredachi Inf., vol. 19, issue 3, pp. 101-102, 1938.

[22] M. Rubio, "Desarrollo orientado a pruebas (TDD)," blog, January 1, 2009; http://altenwald.org/2009/01/08/desarrollo-orientado-a-pruebas-tdd/

[23] P. A. Vaca, C. Maldonado, C. Inchaurrondo, J. Peretti, M. S. Romero, M. Bueno, "Test-Driven Development: Una aproximación para entender su utilidad en el proceso de desarrollo de Software.," Universidad Tecnológica Nacional, Facultad Regional Córdoba.

[24] H. Yanajara, "Manual de SAGE para principiantes," Instituto Técnico de Sonora; http://www.sagemath.org/es/Manual_SAGE_principiantes.pdf

[25] F. Zeng, "Nonlinear codes: Representation, constructions, minimum distance computation and decoding," doctor's degree thesis, Dept. d'Enginyeria de la Informació i de les Comunicacions, Universitat Autònoma de Barcelona, Juny 2014.

[26] Computacional Algebra Group, "Overview of MAGMA V2.17 Features," University of Sidney, January 2016; https://magma.maths.usyd.edu.au/magma/overview/pdf/overv217.pdf

[27] Computacional Algebra Group, "Overview of Magma V2.16: Mathematical Databases," University of Sidney, January 2016 https://magma.maths.usyd.edu.au/magma/overview/2/16/22/

[28] "MATLAB, The Language of Technical Computing," 2016; http://es.mathworks.com/help/matlab/index.html

[29] "MAPLEsoft, Online Help," 2016; http://www.maplesoft.com/support/help/

Victòria González és estudiant de quart de grau en Enginyeria Informàtica a la Universitat Autònoma de Barcelona. El 2015 va col·laborar com a tècnica de suport a la recerca en el Department d'Enginyeria de la Informació i de les Comunicacions (DEIC) i actualment treballa a Accenture.S.L.

APÈNDIXS

A1. APÈNDIX A

APÈNDIX A
CODIS BINARIS ÒPTIMS: RESUM DE PARÀMETRES

		Code				kernel			Span		
Code		n	M	d	$\rho(C)$	M	d	$\rho(k)$	M	d	$\rho(s)$
n4d3M2	Repetition code	4	2	4	2	2	4	2	2	4	2
n5d4M2	Repetition code (extended)	5	2	5	2	2	5	2	2	5	2
n5d3M4	Linear code	5	4	3	2	4	3	2	4	3	2
n6d4M4	Linear code (extended)	6	4	4	3	4	4	3	4	4	3
n6d3M8	Linear code	6	8	3	2	8	3	2	8	3	2
n7d4M8	Linear code (extended)	7	8	4	3	8	4	3	8	4	3
n7d3M16	Hamming code	7	16	3	1	16	3	1	16	3	1
n8d4M16	Hamming code (extended)	8	16	4	2	16	4	2	16	4	2
n8d3M20	NonLinear code	8	20	3	2	1	8	8	256	1	0
n9d4M20	NonLinear code (punctured)	9	20	4	3	1	9	9	512	1	0
n8d5M4	Linear code	8	4	5	3	4	5	3	4	5	3
n9d6M4	Linear code (extended)	9	4	6	4	4	6	4	4	6	4
n9d3M40	Best code (punctured)	9	40	3	2	1	9	9	512	1	0
n10d4M40	Best code	10	40	4	3	1	10	10	512	2	1
n10d3M72	Julin code (72)	10	72	3	2	1	10	10	1024	1	0
n11d4M72	Julin code (72) (extended)	11	72	4	3	1	11	11	1024	2	1
n11d3M144	Julin code (144)	11	144	3	2	2	11	5	2048	1	0
n12d4M144	Julin code (144) (extended)	12	144	4	3	2	12	6	2048	2	1
n11d5M24	Plotkin code (punctured)	11	24	5	3	2	11	5	2048	1	0
n12d6M24	Plotkin code	12	24	6	4	2	12	6	2048	2	1
n12d3M256	Linear code	12	256	3	2	256	3	2	256	3	2
n13d4M256	Linear code (extended)	13	256	4	3	256	4	3	256	4	3
n12d5M32	**Nadler code**	**12**	**32**	**5**	**4**	**4**	**8**	**6**	**256**	**3**	**2**
n13d6M32	**Nadler code (extended)**	**13**	**32**	**6**	**5**	**4**	**8**	**7**	**256**	**4**	**3**
n13d3M512	Linear code	13	512	3	2	512	3	2	512	3	2
n14d4M512	Linear code (extended)	14	512	4	3	512	4	3	512	4	3
n14d3M1024	Linear code	14	1024	3	2	1024	3	2	1024	3	2
n15d4M1024	Linear code (extended)	15	1024	4	3	1024	4	3	1024	4	3
n15d3M2048	Hamming code	15	2048	3	1	2048	3	1	2048	3	1
n16d4M2048	Hamming code (extended)	16	2048	4	2	2048	4	2	2048	4	2
n16d3M2720	Romanov code	16	2720	3	2	32	3	5	65536	1	0
n17d4M2720	Romanov code (extended)	17	2720	4	6	32	4	6	65536	2	1

n17d3M5248	Hamalainen code	17	5248	3	2	32	3	7	131072	1	0
n18d4M5248	Hamalainen code (extended)	18	5248	4	3	32	4	8	131072	2	1
n18d3M10496	Hamalainen code	18	10496	3	2	64	3	2	262144	1	0
n19d4M10496	Hamalainen code (extended)	19	10496	4	3	64	4	7	262144	2	1
n20d9M42	Kaikkonen code	20	42	9	8	1	20	20	1048576	1	0
n21d10M42	Kaikkonen code (extended)	21	42	10	9	1	21	21	1048576	2	1
n21d9M64	**Ostergard code**	**21**	**64**	**9**	**7**	**8**	**11**	**10**	**512**	**4**	**5**
n22d10M64	**Ostergard code (extended)**	**22**	**64**	**10**	**8**	**8**	**12**	**11**	**512**	**4**	**6**
n22d9M80	Ostergard code	22	80	9	8	8	11	11	2048	3	5
n23d10M80	Ostergard code (extended)	23	80	10	9	8	12	12	2048	4	6
n25d9M384	Elssel-Zimmermann code	25	384	9	7	64	11	10	2048	5	5
n26d10M384	Elssel-Zimmermann code (extended)	26	384	10	8	64	12	11	2048	6	6

A2. APÈNDIX B

APÈNDIX B
CONSTRUCCIÓ DE CODIS BINARIS: RESUM DE PARÀMETRES

Best code <length,d,M,dimKernel>	Construction	Best code <length,d,M,dimKernel>	Construction
<12, 3, 40, 2>	DirectSum + Punctured (BBCn9d4M20, BBCn4d3M2)	<17, 6, 48, 4>	DirectSum + Extended (BBCn11d5M24, BBCn5d4M2)
<12, 4, 20, 1>	DirectSum + Shorten (BBCn9d4M20, BBCn4d3M2)	<18, 3, 2304, 32>	DirectSum (BBCn11d3M144, BBcn7d3M16)
<12, 5, 4, 4>	DirectSum + Shorten (BBCn8d5M4, BBCn5d4M2)	<18, 4, 1152, 16>	DirectSum + Extended (BBCn11d3M144, BBcn6d3M8)
<12, 6, 4, 4>	DirectSum + Shorten (BBCn9d6M4, BBCn4d3M2)	<18, 5, 64, 8>	DirectSum (BBCn13d6M32, BBCn5d4M2)
<13, 3, 80, 4>	DirectSum (BBCn8d3M20, BBCn5d3M4)	<18, 6, 64, 8>	DirectSum + Extended (BBCn12d5M32, BBCn5d4M2)
<13, 4, 40, 2>	DirectSum (BBCn9d4M20, BBCn4d3M2)	<19, 3, 2880, 2>	DirectSum (BBCn11d3M144, BBCn8d3M20)
<13, 5, 8, 8>	DirectSum (BBCn8d5M4, BBCn5d4M2)	<19, 4, 2304, 32>	DirectSum + Extended (BBCn11d3M144, BBcn7d3M16)
<13, 6, 4, 4>	DirectSum + Shorten (BBCn9d6M4, BBCn5d4M2)	<19, 5, 96, 8>	DirectSum (BBCn11d5M24, BBCn8d5M4)
<14, 3, 160, 8>	DirectSum (BBCn8d3M20, BBcn6d3M8)	<19, 6, 48, 4>	DirectSum + Shorten (BBCn12d6M24, BBCn8d5M4)
<14, 4, 80, 2>	DirectSum (BBCn10d4M40, BBCn4d3M2)	<20, 3, 5472, 2>	DirectSum + Shorten (BBCn11d3M144, BBCn10d3M72)
<14, 5, 24, 2>	DirectSum + Shorten (BBCn11d5M24, BBCn4d3M2)	<20, 4, 2880, 2>	DirectSum + Extended (BBCn11d3M144, BBCn8d3M20)
<14, 6, 8, 8>	DirectSum + Extended (BBCn8d5M4, BBCn5d4M2)	<20, 5, 128, 16>	DirectSum (BBCn12d5M32, BBCn8d5M4)
<15, 3, 320, 8>	DirectSum (BBCn9d3M40, BBcn6d3M8)	<20, 6, 96, 8>	DirectSum + Extended (BBCn11d5M24, BBCn8d5M4)
<15, 4, 160, 8>	DirectSum + Extended (BBCn8d3M20, BBcn6d3M8)	<21, 3, 10880, 128>	DirectSum (BBCn16d3M2720, BBCn5d3M4)
<15, 5, 32, 4>	DirectSum + Shorten (BBCn12d5M32, BBCn4d3M2)	<21, 4, 5440, 64>	DirectSum (BBCn17d4M2720, BBCn4d3M2)
<15, 6, 24, 2>	DirectSum + Shorten (BBCn12d6M24, BBCn4d3M2)	<21, 5, 128, 16>	DirectSum (BBCn13d6M32, BBcn7d3M16)
<16, 3, 640, 16>	DirectSum (BBCn9d3M40, BBcn7d3M16)	<21, 6, 128, 16>	DirectSum + Extended (BBCn12d5M32, BBCn8d5M4)
<16, 4, 320, 8>	DirectSum + Extended (BBCn9d3M40, BBcn6d3M8)	<22, 3, 21760, 256>	DirectSum + Shorten (BBCn16d3M2720, BBcn7d3M16)
<16, 5, 48, 4>	DirectSum (BBCn11d5M24, BBCn5d4M2)	<22, 4, 10880, 128>	DirectSum + Extended (BBCn16d3M2720, BBCn5d3M4)
<16, 6, 32, 4>	DirectSum + Shorten (BBCn13d6M32, BBCn4d3M2)	<22, 6, 128, 16>	DirectSum (BBCn13d6M32, BBCn9d6M4)
<17, 3, 1152, 16>	DirectSum (BBCn11d3M144, BBcn6d3M8)	<23, 3, 43520, 512>	DirectSum (BBCn16d3M2720, BBcn7d3M16)
<17, 4, 640, 16>	DirectSum + Extended (BBCn9d3M40, BBcn7d3M16)	<24, 4, 43520, 512>	DirectSum + Extended (BBCn16d3M2720, BBcn7d3M16)
<17, 5, 64, 8>	DirectSum (BBCn12d5M32, BBCn5d4M2)	<24, 9, 64, 8>	DirectSum + Shorten (BBCn21d9M64, BBCn4d3M2)
<23, 4, 21760, 256>	DirectSum + Extended (BBCn16d3M2720, BBcn6d3M8)	<24, 10, 42, 1>	DirectSum + Shorten (BBCn21d10M42, BBCn4d3M2)
<23, 6, 128, 16>	DirectSum + Extended (BBCn13d6M32, BBCn9d6M4)	<25, 3, 94464, 64>	DirectSum + Shorten (BBCn17d3M5248, BBCn8d3M20)
<24, 3, 83968, 512>	DirectSum (BBCn17d3M5248, BBcn6d3M8)	<25, 4, 83968, 512>	DirectSum + Extended (BBCn17d3M5248, BBcn6d3M8)

Best code <length,d,M,dimKernel>	Construction	Best code <length,d,M,dimKernel>	Construction
<25, 5, 84, 2>	DirectSum (BBCn20d9M42, BBCn5d4M2)	<30, 4, 1280, 8>	DirectSum (BBCn21d9M64, BBCn9d4M20)
<25, 9, 80, 8>	DirectSum + Shorten (BBCn22d9M80, BBCn4d3M2)	<30, 5, 768, 128>	DirectSum (BBCn25d9M384, BBCn5d4M2)
<25, 10, 64, 8,>	DirectSum + Shorten (BBCn22d10M64, BBCn4d3M2)	<30, 6, 256, 32>	DirectSum (BBCn21d9M64, BBCn9d6M4)
<26, 3, 167936, 1024>	DirectSum (BBCn18d4M5248, BBcn7d3M16)	<30, 10, 384, 64>	DirectSum + Shorten (BBCn26d10M384, BBCn5d4M2)
<26, 4, 167936, 1024>	DirectSum + Extended (BBCn17d3M5248, BBcn7d3M16)	<31, 3, 3072, 512>	DirectSum (BBCn25d9M384, BBcn6d3M8)
<26, 5, 128, 16>	DirectSum (BBCn21d9M64, BBCn5d4M2)	<31, 4, 1600, 8>	DirectSum (BBCn22d9M80, BBCn9d4M20)
<26, 6, 84, 2>	DirectSum+ Extended (BBCn20d9M42, BBCn5d4M2)	<31, 5, 768, 128>	DirectSum (BBCn26d10M384, BBCn5d4M2)
<26, 9, 80, 8>	DirectSum + Shorten (BBCn22d9M80, BBCn5d4M2)	<31, 6, 768, 128>	DirectSum + Extended (BBCn25d9M384, BBCn5d4M2)
<26, 10, 80, 8>	DirectSum + Shorten (BBCn23d10M80, BBCn4d3M2)	<32, 3, 6144, 1024>	DirectSum (BBCn25d9M384, BBcn7d3M16)
<27, 3, 209920, 64>	DirectSum (BBCn18d4M5248, BBCn8d3M20)	<32, 4, 3072, 512>	DirectSum + Extended (BBCn25d9M384, BBcn6d3M8)
<27, 4, 209920, 64>	DirectSum + Extended (BBCn17d3M5248, BBCn8d3M20)	<32, 6, 768, 128>	DirectSum + Extended (BBCn26d10M384, BBCn5d4M2)
<27, 5, 160, 16>	DirectSum (BBCn22d9M80, BBCn5d4M2)	<33, 3, 7680, 64>	DirectSum (BBCn25d9M384, BBCn8d3M20)
<27, 6, 128, 16>	DirectSum + Extended (BBCn21d9M64, BBCn5d4M2)	<33, 4, 6144, 1024>	DirectSum (BBCn25d9M384, BBCn8d4M16)
<27, 10, 80, 8>	DirectSum + Shorten (BBCn23d10M80, BBCn5d4M2)	<33, 5, 1536, 256>	DirectSum (BBCn25d9M384, BBCn8d5M4)
<28, 3, 1024, 128>	DirectSum (BBCn21d9M64, BBcn7d3M16)	<33, 6, 768, 128>	DirectSum + Shorten (BBCn26d10M384, BBCn8d5M4)
<28, 4, 209920, 64>	DirectSum (BBCn18d4M5248, BBCn9d4M20)	<34, 3, 7680, 64>	DirectSum (BBCn26d10M384, BBCn8d3M20)
<28, 5, 168, 4>	DirectSum (BBCn20d9M42, BBCn8d5M4)	<34, 4, 7680, 64>	DirectSum (BBCn25d9M384, BBCn9d4M20)
<28, 6, 160, 16>	DirectSum + Extended (BBCn22d9M80, BBCn5d4M2)	<34, 5, 1536, 256>	DirectSum (BBCn26d10M384, BBCn8d5M4)
<28, 9, 384, 64>	DirectSum + Shorten (BBCn25d9M384, BBCn4d3M2)	<34, 6, 1536, 256>	DirectSum (BBCn25d9M384, BBCn9d6M4)
<29, 3, 1280, 8>	DirectSum (BBCn21d9M64, BBCn8d3M20)	<35, 4, 7680, 64>	DirectSum (BBCn26d10M384, BBCn9d4M20)
<29, 4, 209920, 64>	DirectSum + Extended (BBCn18d4M5248, BBCn9d4M20)	<35, 6, 1536, 256>	DirectSum (BBCn26d10M384, BBCn9d6M4)
<29, 5, 256, 32>	DirectSum (BBCn21d9M64, BBCn8d5M4)	<36, 4, 7680, 64>	DirectSum + Extended (BBCn26d10M384, BBCn9d4M20)
<29, 6, 168, 4>	DirectSum (BBCn20d9M42, BBCn9d6M4)	<36, 6, 1536, 256>	DirectSum + Extended (BBCn26d10M384, BBCn9d6M4)
<29, 9, 384, 64>	DirectSum + Shorten (BBCn25d9M384, BBCn5d4M2)	<52, 3, 786432, 131072>	DirectSum + Punctured (BBCn49d13M393216, BBCn4d3M2)
<29, 10, 384, 64>	DirectSum + Shorten (BBCn26d10M384, BBCn4d3M2)	<52, 13, 393216, 65536>	DirectSum + Shorten (BBCn49d13M393216, BBCn4d3M2)
<30, 3, 1600, 8>	DirectSum (BBCn22d9M80, BBCn8d3M20)	<53, 3, 786432, 131072>	DirectSum + Punctured (BBCn50d14M393216, BBCn4d3M2)

Best code <length,d,M,dimKernel>	Construction	Best code <length,d,M,dimKernel>	Construction
<53, 4, 786432, 131072>	DirectSum (BBCn49d13M393216, BBCn4d3M2)	<56, 6, 786432, 131072>	DirectSum + Extended (BBCn50d14M393216, BBCn5d4M2)
<53, 13, 393216, 65536>	DirectSum + Shorten (BBCn49d13M393216, BBCn5d4M2)	<57, 3, 7864320, 65536>	DirectSum (BBCn49d13M393216, BBCn8d3M20)
<53, 14, 393216, 65536>	DirectSum + Shorten (BBCn50d14M393216, BBCn4d3M2)	<57, 4, 6291456, 1048576>	DirectSum (BBCn49d13M393216, BBCn8d4M16)
<54, 3, 1572864, 262144>	DirectSum (BBCn49d13M393216, BBCn5d3M4)	<57, 5, 1572864, 262144>	DirectSum (BBCn49d13M393216, BBCn8d5M4)
<54, 4, 786432, 131072>	DirectSum (BBCn50d14M393216, BBCn4d3M2)	<57, 6, 786432, 131072>	DirectSum + Shorten (BBCn50d14M393216, BBCn8d5M4)
<54, 5, 786432, 131072>	DirectSum (BBCn49d13M393216, BBCn5d4M2)	<58, 3, 7864320, 65536>	DirectSum (BBCn50d14M393216, BBCn8d3M20)
<54, 14, 393216, 65536>	DirectSum + Shorten (BBCn50d14M393216, BBCn5d4M2)	<58, 4, 7864320, 65536>	DirectSum (BBCn49d13M393216, BBCn9d4M20)
<55, 3, 3145728, 524288>	DirectSum (BBCn49d13M393216, BBcn6d3M8)	<58, 5, 1572864, 262144>	DirectSum + Punctured (BBCn50d14M393216, BBCn9d6M4)
<55, 4, 1572864, 262144>	DirectSum (BBCn49d13M393216, BBcn6d4M4)	<58, 6, 1572864, 262144>	DirectSum + Extended (BBCn49d13M393216, BBCn8d5M4)
<55, 5, 786432, 131072>	DirectSum (BBCn50d14M393216, BBCn5d4M2)	<59, 4, 7864320, 65536>	DirectSum + Extended (BBCn50d14M393216, BBCn8d3M20)
<55, 6, 786432, 131072>	DirectSum + Extended (BBCn49d13M393216, BBCn5d4M2)	<59, 6, 1572864, 262144>	DirectSum + Extended (BBCn50d14M393216, BBCn8d5M4)
<56, 3, 6291456, 1048576>	DirectSum (BBCn49d13M393216, BBcn7d3M16)	<60, 4, 7864320, 65536>	DirectSum + Extended (BBCn50d14M393216, BBCn9d4M20)