Conception Avancée des codes LDPC binaires pour des applications pratiques

Madiagne Diouf

Université de Cergy-Pontoise
École doctorale Sciences & Ingénierie

Université de Cheikh Anta DIOP
École doctorale Mathématiques & Informatique

THÈSE

en Cotutelle

présentée en vue d'obtenir le grade de

DOCTEUR

spécialité Sciences et Technologies
de l'Information et de la Communication

par

Madiagne DIOUF

Conception Avancée des codes LDPC binaires pour des applications pratiques

Soutenue le 07 Décembre 2015
devant la Commission d'Examen

Jury :

Rapporteur	Professeur	Charly POULLIAT	INP-ENSEEIHT, Toulouse
Rapporteur	Professeur	Jean-Pierre CANCES	XLIM, Université Limoges
Président	Professeur	Emmanuel BOUTILLON	Lab-STICC, UBS, Lorient
Examinateur	Maître de Conférences	Alban GOUPIL	SysCom-CReSTIC, Reims
Examinateur	Professeur agrégé en mathématiques	Philippe BOUAFIA	ETIS-ENSEA, Cergy
Directeur de thèse	Professeur	David DECLERCQ	ETIS-ENSEA, Cergy
Co-directeur de thèse	Docteur	Samuel OUYA	LIRT-ESP, Dakar

Remerciements

Je tiens à adresser de vifs et sincères remerciements à mon directeur de thèse David DECLERCQ. Sa disponibilité, sa générosité intellectuelle, son dynamisme, son objectivité, ses remarques pertinentes, son soutien social et la confiance qu'il a sue m'accorder m'ont permis de mener à bien mes travaux de thèse. Encore merci. Je remercie également mon co-directeur de thèse Samuel OUYA avec qui j'ai débuté la recherche. Il m'a toujours donné d'excellents conseils, et n'a jamais cessé de m'encourager.

Je remercie ensuite les rapporteurs de cette thèse Charly POULLIAT et Jean-Pierre CANCES. Leurs remarques et leurs commentaires m'ont aidé à améliorer le contenu de ce document. Je tiens à remercier Emmanuel BOUTILLON, qui m'a fait l'honneur de présider le jury. Je remercie également Alban GOUPIL et Philippe BOUAFIA d'avoir accepté de faire parti du jury de soutenance de cette thèse.

Je remercie Mathias QUOY directeur du laboratoire ETIS où j'ai effectué une partie de cette thèse. Merci également à Inbar Fijalkow ancienne directrice du laboratoire de m'avoir accueilli au sein du laboratoire ETIS. Mes remerciements à l'endroit de tous les doctorants et aux membres du laboratoire particuliérement à Annick Bertinotti pour ses aides dans toutes les démarches administratives et sociales. Je ne peux m'empêcher de citer les doctorants avec qui j'ai partagé le bureau 356 ; Christiane Kameni-Ngassa, Mohamed, Fréderic, Khao pour les cafés et Alexandre le nouveau à qui je dis bon courage.

Je remercie tous les membres du laboratoire LIRT de Dakar, Sénégal où j'ai fait une grande partie de mes travaux lors de cette thèse. Merci beaucoup à Mamadou CAMARA et à Césaire NDIAYE pour vos lectures. Je vous en suis très reconnaissant. Je tiens à remercier particulièrement Abdourahmane NDIAYE avec qui j'ai passé beaucoup de temps à discuter sur mes travaux. Il a toujours su m'accorder son écoute et me faire des remarques pertinentes.

Je remercie Marc Fossorier, nos multiples rencontres ont été riche en échange scientifique et m'ont beaucoup aidé sur une partie de mes travaux.

Je remercie tous mes amis qui m'ont toujours encouragé dans mes études en particulier Pape Ibrahima NDOUR qui m'a fait aimer les mathématiques et Massanga DIONGUE qui a toujours été proche. Merci à Lamine DIOP, Ibrahim SALL, Cheikh LÔ mon maître coranique, Ibra DIOP et Aliou DIOUF qui ont rendu mes séjours agréables. Merci à Cheikh Malick FALL et à sa famille ; je n'ai jamais été seul durant les fêtes.

Je remercie tous les enseignants qui ont contribué à ma formation, particulièment à Aicha KONE. Merci à Serigne Hamsatou Mourtada MBACKE qui m'a initié sur la spritualité et m'a toujours encouragé dans la recherche de la connaissance.

Je ne pourrais vous remercier assez, pour votre patience, votre soutien et vos conseils. Merci beaucoup à toute ma famille et ma belle-famille, en particulier à Assane BEYE, Maguette DIOUF, Mame Lesse DIOUF et Pierre DIOUF. Mention spéciale à mon père, Papa tu t'es investi corps et âme pour la réussite de tes enfants. Je te dois tout !

Je ne saurais comment te remercier, Yague ma femme, ma force. Tu as fait preuve de patience, de compréhension et de soutien immesurable durant ces dernières années. Tu as supporté la solitude durant mes séjours en FRANCE, tu m'as toléré mes absences pendant des moments importants de la vie de notre petite fille. Tu m'as toujours encouragé à continuer au moment où je voulais arrêter. Tu m'as félicité pour le moindre résultat que tu rendais si important en me disant : "sois fier de toi". Ce travail est le tien et j'espère que tu en seras fière. Une fois de plus merci pour tout.

A toute ma famille,
A ma femme Yague GUEYE DIOUF,
A ma fille Ndèye-Sokhna DIOUF,
A ma mère Ndéye-Sokhna SECK, paix à son âme

Résumé

La conception de codes LDPC binaires avec un faible plancher d'erreurs est encore un problème considérable non entièrement résolu dans la littérature. Cette thèse a pour objectif la conception optimale/optimisée de codes LDPC binaires. Nous avons deux contributions principales pour la construction de codes LDPC à faible plancher d'erreurs. Notre première contribution est un algorithme qui permet de concevoir des codes QC-LDPC optimaux à large girth avec les tailles minimales. Nous montrons par des simulations que notre algorithme atteint les bornes minimales fixées pour les codes QC-LDPC réguliers $(3, d_c)$ avec d_c faible. Notre deuxième contribution est un algorithme qui permet la conception optimisée des codes LDPC réguliers en minimisant les trapping-sets/expansion-sets dominants(es). Cette minimisation s'effectue par une détection prédictive des trapping-sets/expansion-sets dominants(es) définies pour un code régulier $\mathcal{C}(d_v, d_c)$ de girth g_t. Par simulations sur des codes de rendement différent, nous montrons que les codes conçus en minimisant les trapping-sets/expansion-sets dominants(es) ont de meilleures performances que les codes conçus sans la prise en compte des trapping-sets/expansion-sets. Les algorithmes que nous avons proposés se basent sur le RandPEG généralisé. Ces algorithmes prennent en compte les cycles non-vus dans le cas des codes quasi-cycliques pour garantir les prédictions.

Mots-clés : code LDPC, graphe de Tanner, cycle, girth, décodeur itératif, plancher d'erreurs, Progressive Edge-Growth (PEG), code Quasi-Cyclique (QC), Matrice de permutation circulante, trapping-sets.

Abstract

The design of binary LDPC codes with low error floors is still a significant problem not fully resolved in the literature. This thesis aims to design optimal/optimized binary LDPC codes. We have two main contributions to build the LDPC codes with low error floors. Our first contribution is an algorithm that enables the design of optimal QC-LDPC codes with maximum girth and mini-

mum sizes. We show by simulations that our algorithm reaches the minimum bounds for regular QC-LDPC codes $(3, d_c)$ with low d_c. Our second contribution is an algorithm that allows the design optimized of regular LDPC codes by minimizing dominant trapping-sets/expansion-sets. This minimization is performed by a predictive detection of dominant trapping-sets/expansion-sets defined for a regular code $\mathcal{C}(d_v, d_c)$ of girth g_t. By simulations on different rate codes, we show that the codes designed by minimizing dominant trapping-sets/expansion-sets have better performance than the designed codes without taking account of trapping-sets/expansion-sets. The algorithms we proposed are based on the generalized RandPEG. These algorithms take into account non-cycles seen in the case of quasi-cyclic codes to ensure the predictions.

Keywords : LDPC code, Tanner graph, cycle, girth, iterative decoder, error floor, Progressive Edge-Growth (PEG), Quasi-Cyclic (QC) code, Circulant Permutation Matrix, trapping-sets.

Table des matières

Table des figures

Liste des tableaux

Glossaire

ACE	Approximate Cycle EMD.
AS	Absorbing Sets .
AWGN	Additive White Gaussian Noise.
BEC	Binary Erasure Channel.
BER	Bit Error Rate.
BIBD	Balanced Incomplete Block Design.
BP	Belief Propagation.
BPSK	Binary Phase Shift Keying.
BSC	Binary Symmetric Channel.
CBE	Canal Binaire à Effacement.
CBS	Canal Binaire Symétrique.
EbNo	Energie sur Bruit.
EMD	Extrinsic Message Degree.
FER	Frame Error Rate.
LDPC	Low Density Parity Check.
MPC	Matrice de Permutation Circulante.
QC	Quasi-Cyclic.
PEG	Progressive Edge Growth.
RA	Repeat Accumulate.
RSB	Rapport Signal à Bruit.
SNR	Signal to Noise Rate .
SPA	Sum Product Algorithm.
SS	Stopping Sets.
TEB	Taux d'Erreur Binaire.
TS	Trapping Sets.
TSE	Trapping Sets Elémentaire.

Introduction

Historique et Motivations

Depuis 1948, le monde des télécommunications a connu un véritable développement dû aux résultats théoriques établis par Shannon [Sha48] et aux développement de l'électronique. L'un des résultats établi par Shannon est : *"Dans une transmission numérique en présence de perturbation, si le niveau moyen de celle-ci ne dépasse pas un certain seuil de puissance et en utilisant un* **codage approprié***, le récepteur peut identifier le message d'origine sans aucune erreur"*. Ce résultat a depuis constitué pour les chercheurs et ingénieurs un défi scientifique qui consiste à trouver le codage approprié.

Les codes LDPC (Low Density Parity-Check) forment une classe de codes en bloc qui se caractérisent par une matrice de contrôle creuse. Ils ont été décrits pour la première fois dans la thèse de Gallager [Gal62] au début des années 60, puis redécouverts par MacKal [MN96] *et al*, Wiberg [Wib96] et Sipser *et al.* [SS94]. Les travaux de MacKay et Neal [MN96, Mac99], montrent que les codes LDPC peuvent s'approcher des performances limites de Shannon. Chung *et al* [CFRU01] montrent que la capacité des codes LDPC est seulement à 0.0045dB de la limite de Shannon, ce qui en fait le meilleur code de correction d'erreurs connu jusqu'ici.

Ainsi au cours de ces deux dernières decénnies plusieurs méthodes de conception de *bons codes* LDPC ont été développées [Mac99, RSU01, RU08, LMSS01, CFRU01, HEA01, HEA02, HEA05, KLF01]. Un *bon code* dépend fortement des contraintes fixées suivant l'application pour laquelle le code sera utilisé. Ces contraintes sont sa performance de correction d'erreurs et sa complexité d'implémentation.

∗ En termes de *performance de correction d'erreurs*, nous avons principalement deux critères :

Perf.1 *Bonnes performances dans la zone de convergence ("Good water-fall")* : c'est à dire obtenir le plus faible rapport signal sur bruit (RSB) pour des taux d'erreurs trame autour de 10^{-2} à 10^{-3}. C'est le cas typique des communications sans fils de type téléphonie (3G, 3G+, 4G).

Perf.2 *Faible plancher d'erreurs ("Low error-floor")* : c'est à dire obtenir un taux d'erreurs faible pour un SNR pouvant assurer une transmission dite *quasi-error free*. Les taux d'erreurs souhaités peuvent être de l'ordre de 10^{-7} à 10^{-9} suivant le type d'application visée. Les applications utilisant des taux d'erreurs si bas comprennent les communications satellitaires, les systèmes de communications par fibre optique ainsi que les enregistrements magnétiques [XZKB10].

Une illustration de ces critères sous forme de région est faite sur la figure 1.

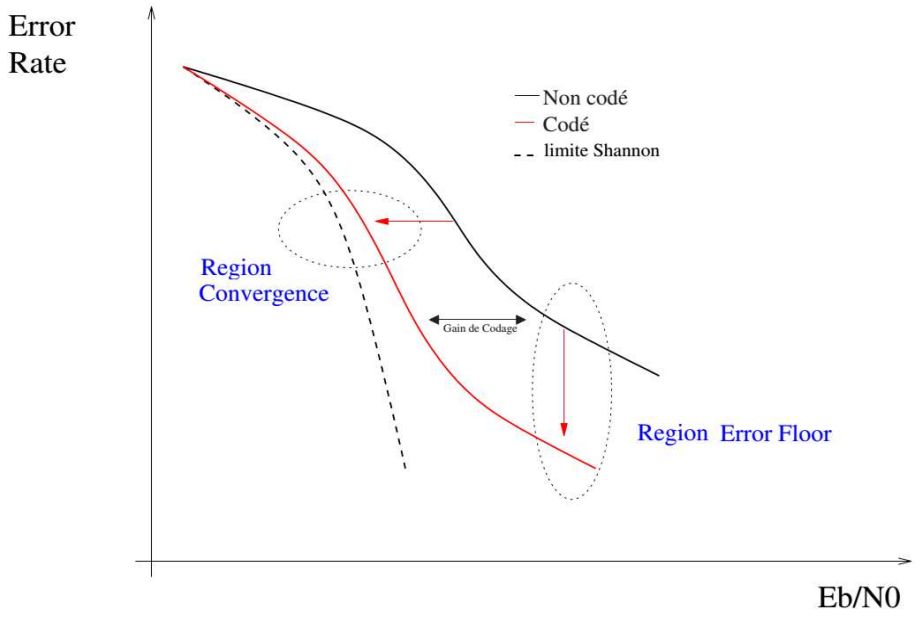


FIGURE 1 – Les régions caractérisant les performances d'un système de codage

∗ En termes de complexité d'implémentation des fonctions de codage et de décodage, il y'a trois critères à prendre en compte :

Comp.1 L'*encodage linéaire rapide*, en principe l'encodage des codes LDPC a une compléxité quadratique avec la taille du code.

Comp.2 *Haut facteur de parallélisme pour le décodeur*, puisque généralement, c'est le décodage qui présente la complexité dominante des codes LDPC.

Comp.3 Une *faible latence*, c'est-à-dire le plus petit nombre d'itérations de décodage pour atteindre les performances voulues.

Des études théoriques permettent de traiter les critères **Perf.1, Comp.1, Comp.2** et **Comp.3**, et de les relier à des propriétés structurelles des codes LDPC. Par exemple, l'utilisation des structures dites en zig-zag, dans la partie redondante des matrices de parité, permet de satisfaire la contrainte d'implémentation **Comp.1**. La conception de code LDPC basée sur les protographes permet de satisfaire la contrainte **Comp.2**, de plus il est possible de construire des codes LDPC basés sur les protographes et les structures zig-zag telles que les contraintes **Comp.3** et **Perf.1** soient très satisfaites.

Cependant, très peu de travaux s'interessent à la réalisation de la contrainte **Perf.2**, à savoir la conception de codes à faible plancher d'erreurs qui est la plus difficile à obtenir et est de plus en plus nécessaire dans les applications pratiques. C'est récemment qu'il est bien connu que le plancher d'erreurs des décodeurs itératifs LDPC est dû à la présence de certaines topologies dans le graphe de Tanner [Ric03]. Ces topologies sont appelées *"trapping-sets"* et sont formées par une combinaison de plusieurs cycles courts dans le graphe de Tanner. Il va de soi que la conception de codes LDPC binaires avec un faible plancher d'erreurs nécessite alors l'élimination des trapping-sets. Cependant la plupart des méthodes proposées dans la littérature ne permettent pas une élimination prédictive des trapping-sets. *Ainsi, l'objectif visé dans cette thèse est le développement d'outils automatiques pour la construction optimale/optimisée de codes LDPC.*

L'algorithme PEG est une méthode très efficace et bien connu pour construire un graphe de Tanner avec de long cycle, mais il est "greedy" dans sa version originale. Une de ses versions améliorées "non-greedy" est l'algorithme RandPEG où l'arbre des voisins est tronqué en fonction du girth souhaité et que le nombre de cycle court crée est minimisé. Notre outil principal sera cet algorithme RandPEG dont nous proposons une extension à des constructions plus avancées. Nous nous focalisons plus sur les codes LDPC quasi-cycliques (QC-LDPC) et les protographes qui sont très utilisés dans les applications pratiques telles que la norme DVB-S2 ou les normes 802.16e et 802.11n.

Dans cette thèse nous avons trois contributions principales. Les deux premières contributions sont décrites dans le chapitre 2, la première consiste à la définition d'une nouvelle borne atteignable pour les codes QC-LDPC et la deuxième consiste à la conception optimale de code QC-LDPC avec les girths maximums et les tailles minimales. La troixième contribution est la conception optimisée de codes réguliers LDPC (QC, protographes) en minimisant les trapping-sets/expansion-sets dominants.

Organisation du manuscrit

Ce manuscrit est composé de trois chapitres organisés comme suit :

Le ***chapitre 1*** est consacré à l'état de l'art sur la conception des codes LDPC. Nous faisons un rappel de quelques définitions et notations pour les codes LDPC, suivi d'une brève description générale sur les méthodes de constructions algébriques et les méthodes de construction pseudo-aléatoires.

Le ***chapitre 2*** présente nos deux premières contributions majeures. En effet, dans l'état de l'art des codes quasi-cyclique réguliers les bornes minimales proposées pour les matrices de permutation circulante ne sont pas "tight" (ie. atteignables). Or pour définir l'optimalité de la conception d'un code il est impératif de savoir les bornes "tight". Ainsi nous proposons dans ce chapitre une première contribution qui est une amélioration des bornes existantes pour les codes QC-LDPC. Ensuite nous proposons une deuxième contribution qui est un algorithme généralisé du RandPEG. Cet algorithme atteint les bornes minimales pour les faibles tailles et prend en considération les *cycles non-vus* lors de la construction des codes QC-LDPC à large *girth*.

Le ***chapitre 3*** présente notre troixième contribution majeure qui est un algorithme de construction de code LDPC sans trapping-set/expansion-sets. La particularité de cet algorithme est que la détection des topologies, trapping-sets ou expansion-sets, se fait de manière prédictive à partir de l'arbre des voisins en tenant compte des cycles non-vus pour les codes QC-LDPC. Par conséquent nous n'avons pas besoin de procéder à des vérifications après la création d'une nouvelle branche à la différence des méthodes existantes où une vérification est obligatoire. Dans ce chapitre nous avons fait en premier temps une étude détaillée des topologies particulières (cycles, trapping-sets, stopping-sets, expansion-sets) présentes dans le graphe de Tanner. Puis nous présentons les critères de détection de ces structures particulières à partir d'un arbre des voisins et enfin nous décrivons les algorithmes RandPEG modifiés. Pour chaque algorithme proposé, des comparaisons de statistiques et de courbes de performances sont effectuées.

Finalement, nous conclurons notre travail par une synthèse des différentes solutions proposées pour la conception optimale et optimisée de code LDPC avec un faible plancher d'erreurs pour finir par des perspectives qui seront aussi décrites.

Chapitre 1

Etat de l'art sur la conception de codes LDPC (Low Density Parity-Check)

Dans ce chapitre nous présentons l'historique de la conception des codes LDPC, puis nous donnons quelques définitions et notations qui seront utilisées dans ce manuscrit, et nous terminons par un tour d'horizon sur les techniques de conception de codes LDPC.

1.1 Historique de conception de codes LDPC

Dans les systèmes de communication numérique, le message transmis quitte une source (émetteur) vers une destination (récepteur) en passant par une voie appelée canal de transmission. Ce canal n'est pas parfait, il présente des parasites qui peuvent induire à des erreurs sur le message qui sera reçu. Shannon a démontré dans [Sha48] que :

Théorème 1.1 *Tout canal de transmission admet un paramètre C, appelé capacité du canal, tel que pour tout $\epsilon > 0$ et pour tout $R < C$, il existe un code de taux R permettant la transmission du message avec un taux d'erreurs blocs inférieur ou égal à ϵ.*

Cependant, il n'a pas souligné les méthodes d'élaboration de ce codeur. Depuis lors plusieurs codes correcteurs ont été développés dont les plus performants sont ceux qui utilisent des algorithmes itératifs. Parmi ces codes nous avons les Turbo-codes qui sont introduits par Berrou et al. [BGT93] et les codes LDPC introduits par Gallager [Gal62] qui sont les plus présents dans les systèmes de communication

numérique actuels. Ces codes font l'objet permanent de perfectionnement pour se rapprocher des limites de Shannon. Dans cette thèse nous nous limitons aux codes *LDPC* qui présentent une plus faible complexité par rapport aux Turbo-codes.

Les codes LDPC se caractérisent par une matrice de contrôle creuse c'est-à-dire à faible densité d'où le nom LDPC (Low-Density Parity-Check) donné à ce codage. Ces codes sont introduits en 1960 par Gallager qui proposa un algorithme itératif pour la correction des erreurs à la réception. Cet algorithme est appelé *Gallager A/B* et il consiste à chercher les bits les plus susceptibles d'être erronés et de les inverser lors de chaque itération. Gallager montre que si la position des éléments non nuls de la matrice de contrôle ne forme pas de *"cycle"* alors la probalité d'erreur s'approche de zéro pour une augmentation des itérations. Mais l'apparition des codes de Reed-Solomon, en cette même année, a fait que les codes LDPC étaient oubliés. Quelques rares études y font référence durant cette période de sommeil, notamment celle de Margulis sur des constructions sans cycle court [Mar82,Mar88] et celle de Tanner [Tan81] qui proposa une généralisation des codes de Gallager et une représentation par graphe biparti. C'est après l'invention des Turbo-codes que les codes LDPC furent redécouverts au milieu des années 90 par MacKal [MN96] *et al*, Wiberg [Wib96] et Sipser *et al.* [SS94]. Les travaux de MacKay et Neal [MN96, Mac99], montrent que les codes LDPC, peuvent s'approcher des performances limites de Shannon avec certaintes contraintes sur la matrice de contrôle. Chung *et al* [CFRU01] montrent que la capacité des codes LDPC est seulement à $0.0045dB$ de la limite de Shannon tandis que les Turbo-code sont à $0.7dB$ [BGT93], ce qui fait des codes LDPC les meilleurs codes de correction d'erreurs connus jusqu'ici. Toutefois, ces résultats théoriques sur les performances des codes LDPC supposent que les positions de tous les éléments non nuls de la matrice ne forment pas de cycle. Ainsi, construire un code LDPC revient à définir la position de tous les éléments non nuls dans la matrice de contrôle de parité. Avant d'aborder l'étude plus détaillée des méthodes et algorithmes de construction, nous donnons d'abord quelques définitions et notations liées aux codes LDPC.

1.2 Définitions et Notations

Un code LDPC est caractérisé par sa matrice de contrôle creuse ; c'est-à-dire une matrice dont le nombre d'éléments non nuls est très faible par rapport au nombre d'éléments nuls. Si tous les élément non nuls valent "1" alors le code est dit *binaire* sinon le code est dit *non-binaire*. Dans cette thèse nous nous restreignons aux codes binaires. Un code LDPC binaire noté $\mathcal{C}(N, K)$, où N représente la taille des mots codes et K le nombre de bits d'informations, est de rendement $R = K/N$

et possède deux formes de représentation : une matricielle et une autre graphique.

1.2.1 Représentation matricielle

La matrice de contrôle creuse est notée H et elle est de dimension $M \times N$, où $M \geq N - K$ est le nombre de bits redondants. Le coefficient de H à la $(m+1)^{eme}$ ligne et $(n+1)^{eme}$ colonne est notée $h_{m,n}$ et elle vaut "1" ou "0". Le nombre de "1" par ligne et le nombre de "1" par colonne sont leurs poids respectifs. Si toutes les lignes ont le même poids d_c alors le code est dit *régulier en ligne*, si toutes les colonnes ont le même poids d_v alors le code est dit *régulier en colonne*. Un *code régulier* est un code régulier en ligne et en colonne, voir exemple 1.1. Dans le cas contraire, le code est *irrégulier*. Un code régulier est noté par $\mathcal{C}(N, d_v, d_c)$ ou code régulier (d_v, d_c).

Exemple 1.1 *Code régulier avec $d_v = 2$ et $d_c = 3$.*

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \tag{1.1}$$

1.2.2 Représentation graphique

Un code LDPC peut également être représenté sous une forme graphique. Cette représentation est appelée *graphe biparti ou graphe de Tanner* [Tan81]. Ce graphe noté G est constitué d'un ensemble de N *nœuds de variable* notés $V = \{v_0, v_1, \cdots, v_{N-1}\}$, et d'ensemble de M *nœuds de contrôle* notés $C = \{c_0, c_1, \cdots, c_{M-1}\}$. Par convention, sur le graphe biparti les nœuds de variable seront représentés par des cercles "◯" et les nœuds de contrôle par des carrés "□". Un nœud de variable v_n et un nœud de contrôle c_m sont reliés par une *branche* si $h_{m,n} = 1$ et dans ce cas les deux nœuds sont dits *voisins*. Le *degré* d'un nœud dans G est son nombre de voisins dans G. Un code $\mathcal{C}$ représenté par le graphe G est dit de *poids colonne régulier* d_v si tous les nœuds de variable dans V ont le même degré d_v, le graphe G est dit de *poids ligne régulier* si tous les nœuds de contrôle dans C ont le même degré d_c. Le code régulier est noté code LDPC régulier (d_v, d_c). Nous donnons dans la suite quelques définitions particulières pour un graphe.

Définition 1.1 *Un **chemin** dans G est une séquence alternée de nœuds de variable et nœuds de contrôle distincts $v_{n_0}, c_{m_0}, v_{n_1}, c_{m_1}, \ldots, v_{n_k}$, tel que v_{n_i}, et c_{m_i}, sont des voisins pour $1 \leq i \leq k$. Deux chemins sont distincts s'ils diffèrent au moins d'un nœud.*

Définition 1.2 *Un l**-cycle** ou cycle de longueur l dans G est une suite alternatée de nœuds de variable et de nœuds de contrôle $v_{n_0}, c_{m_0}, v_{n_1}, c_{m_1}, \ldots, v_{n_{l/2}}$, avec $v_{n_0} = v_{n_{l/2}}$. Par conséquent dans un graphe biparti G, la longueur d'un cycle l est paire.*

Définition 1.3 *Le **girth** d'un graphe G est la longueur du plus petit cycle dans G, il est noté g.*

Par exemple, la figure 1.1 est la représentation graphique de la matrice de contrôle définie dans l'exemple 1.1, le girth est $g = 6$. Le chemin (v_1,c_1,v_3,c_4,v_5,c_2,v_1) en rouge décrit un cycle de longueur 6.

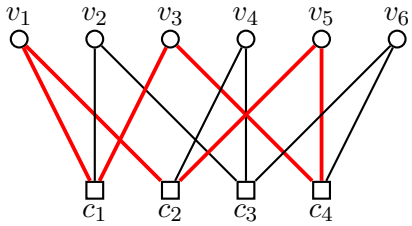


FIGURE 1.1 – Graphe de Tanner du code défini dans l'exemple 1.1, $d_v = 2$,$d_c = 3$

Les cycles ont un impact considérable sur un décodeur itératif.

Par exemple, pour un graphe sans cycle l'algorithme de propagation de croyance (BP pour Belief Propagation) converge vers la solution du maximum de vraissemblance. Par contre, pour un graphe avec des cycles l'algorithme de propagation de croyance n'est pas optimal. En effet, entre deux itérations successives, la présence de cycles dans le graphe G entraine une rupture de l'hypothèse d'indépendance des fonctions de parités dans l'algorithme de propagation de croyance. Cette rupture engendre une dégradation des performances de décodage.

Et de manière générale, Gallager a montré d'après le lemme suivant que le nombre d'itérations indépendantes dépend du girth g et est égal à $N_{iter} = \left\lceil \frac{g-4}{4} \right\rceil$.

Lemme 1.1 *Si un code a un girth g, alors le nombre d'itérations indépendantes N_{iter} satisfait à :*

$$N_{iter} < \frac{g}{4} \leq N_{iter} + 1 \tag{1.2}$$

Par conséquent, la conception de code LDPC avec un graphe qui ne présente pas de cycle est l'idéal pour que les itérations soient indépendantes mais c'est impossible dans la pratique. Ainsi plusieurs travaux se sont interessés sur la conception de codes LDPC dans le but d'augmenter la taille du girth ou de minimiser le nombre de cycles courts. Ces travaux sont classés en deux grandes familles principales, les constructions structurées et les constructions pseudo-aléatoires.

1.3 Les constructions structurées

Plusieurs méthodes de constrution structurées ont été proposées. Ces méthodes sont basées soit sur les *matrices de permutation*, soit sur les *géométries finies*, soit les combinatoires.

1.3.1 Construction basée sur les Matrices de Permutation

Avant d'aborder la describtion des différentes méthodes de conception de code LDPC basées sur les matrices de permutation, nous donnons quelques définitions.

Définition 1.4 *Une* ***matrice de permutation*** *est une matrice carrée tel que :*

- *les coefficients sont "0" ou "1",*
- *chaque ligne a un et un seul "1",*
- *chaque colonne a un et un seul "1"*

Exemple 1.2 *La matrice identité est un cas particulier de matrice de permutation où les "1" sont sur la diagonale de la matrice carrée. Par exemple la matrice identité d'ordre 5 est sous la forme :*

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Définition 1.5 *Une* ***matrice de permutation circulante*** *est une matrice de permutation dans laquelle on passe d'une colonne à la suivante par décalage vers le bas.*

Cette définition peut être faite avec les lignes et on dit qu'on passe d'une ligne à la suivante par décalage à droite. Ainsi une matrice de permutation circulante peut être interprétée comme une matrice identité $p \times p$ décalée de p_i vers le bas ou vers la gauche suivant la définition utilisée. p_i est l'indice de la ligne à coefficient "1"

pour la première colonne et $p = N/d_c$ est le facteur de décalage. Ainsi pour une colonne donnée c, $0 \leq c \leq p-1$, nous avons 1 à la ligne $r = c + p_{j,l}\ mod\ p$ et 0 ailleurs. Nous utilisons la notation I_{p_i} pour désigner une matrice de permutation circulante dont le décalage vers le bas est de p_i par rapport à la matrice identité.

Exemple 1.3 *Pour une matrice d'ordre 5, I_3 est sous la forme :*

$$I_3 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Du fait de leurs configurations, les matrices de permutation circulante sont très favorables dans l'implémentation matérielle des codes LDPC. Elles sont utilisées dans plusieurs méthodes de conception de code que nous décrivons ici.

• Les codes de Gallager

Gallager initie une première méthode de construction de code LDPC régulier [Gal63, annexe C]. Cette méthode de construction qui se base sur les matrices de permutation qui ne sont pas forcément circulaires. Le processus de construction des codes réguliers $\mathcal{C}(N, d_v, d_c)$ s'effectue comme suit :

- d'abord une sous-matrice de permutation H_1 de dimension $N/d_c \times N$ est créée, telles que les d_c premières colonnes de la première ligne sont à "1" et chacune des lignes restantes de H_1 est obtenue par décalage vers la droite de d_c par rapport à la ligne précédente.
- puis les $d_v - 1$ sous-matrices sont créées par des permutations aléatoires sur H_1

Exemple 1.4 *Une matrice de contrôle H d'un code LDPC régulier $\mathcal{C}(12, 3, 4)$ avec la méthode de Gallager est :*

$$H = \left[\begin{array}{cccccccccccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ \hline 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ \hline 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{array}\right] \quad (1.3)$$

Cette méthode de construction proposée par Gallager ne garantit pas un large girth qui est un critère important pour l'optimalité d'un décodeur itératif. Durant les années 2000, de nouvelles méthodes de conception basées toujours sur les matrices de permutation ont vu le jour. Parmi ces méthodes nous avons les *array codes*, les *codes bloc QC LDPC*, les protographes et les *codes Repeat Accumulate*. Une brève description de ces méthodes est proposée.

• Les "Array Codes"

Les *array codes* sont proposés par Fan dans [Fan00]. La matrice de contrôle d'un array code LDPC régulier est caractérisée par trois paramètres (p, d_v, d_c) où p est un nombre premier, tel que $d_v < d_c \leq p$. Cette matrice de dimension $p.d_v \times p.d_c$ est sous la forme :

$$H_a = \begin{bmatrix} I & I & \cdots & I & \cdots & I \\ I & P^1 & \cdots & P^{d_v-1} & \cdots & P^{d_c-1} \\ I & P^2 & \cdots & P^{2(d_v-1)} & \cdots & P^{2(d_c-1)} \\ \vdots & \vdots & & \vdots & & \vdots \\ I & P^{d_v-1} & \cdots & P^{(d_v-1)(d_v-1)} & \cdots & P^{(d_v-1)(d_c-1)} \end{bmatrix} \tag{1.4}$$

où I est la matrice identité de taille $p \times p$ et P une matrice de permutation circulante différente de I. Avec la condition $I \neq P$, la matrice H_a ne contient pas de cycle de longueur 4. Ces modèles de matrice sont utilisés dans la conception de codes LDPC par l'approche *"shortening"* [MKL06] pour avoir des codes de large girth. L'approche "shortening" consiste à effectuer des observations sur une matrice de dimension plus importante puis d'éliminer les cycles courts par suppression des blocs colonnes correspondants aux nœuds de variable présents dans beaucoup de cycles courts. Dans [MKL06], Milenkovic et al. définissent d'abord les conditions d'existence d'un 6-cycle à partir des indices des blocs colonnes de la matrice H_a pour les codes de poids colonne 3 et 4 avec $d_c = p$. Puis ils montrent que ces codes de matrice H_a sont de girth supérieure où égale à 8 pour $d_v \geq 3$

• Les Codes bloc QC LDPC

La matrice de parité H d'un code Quasi-Cyclique LDPC régulier noté QC-LDPC régulier (d_v,d_c) de longueur $N = p.d_c$ peut être représentée sous la forme :

$$H = \begin{bmatrix} I_{p_{0,0}} & I_{p_{0,1}} & \cdots & I_{p_{0,d_c-1}} \\ I_{p_{1,0}} & I_{p_{1,1}} & \cdots & I_{p_{1,d_c-1}} \\ \vdots & \vdots & \ddots & \vdots \\ I_{p_{d_v-1,0}} & I_{p_{d_v-1,1}} & \cdots & I_{p_{d_v-1,d_c-1}} \end{bmatrix} \tag{1.5}$$

où $I_{p_{i,j}}$ est la matrice de permutation circulante de dimension $p \times p$. Cette architecture de H, avec les bloc-circulants, fait que les codes QC-LDPC sont encodés en temps linéaire avec des registres de décalage et nécessitent peu d'espace mémoire pour stocker les graphes de code pour le décodage. En effet, il suffit de mémoriser les premières rangées de chaque sous-matrice, les autres rangées sont formées par permutations circulaires. Par conséquent, les codes QC-LDPC sont adaptés pour la mise en œuvre matérielle.

Tanner et al proposent dans [TSS+04] deux modèles de construction de codes avec des girth $g \geq 8$. Il s'agit de la construction de codes LDPC convolutional et de la construction de codes bloc QC-LDPC dont nous faisons une brève description. L'approche construction de codes bloc QC LDPC est celle des expansions de graphe introduites comme un outil d'analyse en théorie de codage par Sipser et Spielman dans [SS94]. La construction du code LDPC par expansion consiste à définir un graphe de base associé à une matrice de contrôle de parité de base. L'expansion de cette matrice est réalisée en remplaçant chaque élément non nul de la matrice par une matrice d'expansion. Dans le cas de Tanner et al [TSS+04], la matrice de contrôle de base est sous la forme :

$$H_B = \begin{bmatrix} 1 & a & a^2 & \cdots & a^{d_c-1} \\ b & ba & ba^2 & \cdots & ba^{d_c-1} \\ b^2 & b^2a & b^2a^2 & \cdots & b^2a^{d_c-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ b^{d_v-1} & b^{d_v-1}a & b^{d_v-1}a^2 & \cdots & b^{d_v-1}a^{d_c-1} \end{bmatrix} \tag{1.6}$$

La matrice de contrôle H est obtenue en remplaçant chaque coefficient $h_{B_{i,j}}$ de H_B par la matrice de permutation circulante $I_{b^i a^j}$. La matrice H est alors sous la forme :

$$H = \begin{bmatrix} I_1 & I_a & I_{a^2} & \cdots & I_{a^{d_c-1}} \\ I_b & I_{ba} & I_{ba^2} & \cdots & I_{ba^{d_c-1}} \\ I_{b^2} & I_{b^2a} & I_{b^2a^2} & \cdots & I_{b^2a^{d_c-1}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ I_{b^{d_v-1}} & I_{b^{d_v-1}a} & I_{b^{d_v-1}a^2} & \cdots & I_{b^{d_v-1}a^{d_c-1}} \end{bmatrix} \tag{1.7}$$

Si d_v et d_c divisent $p-1$ avec p est premier, alors $o(a) = d_c$ et $o(b) = d_v$. Un

exemple est la matrice référence de girth $g = 8$ construite avec les paramètres : $p = 31$, $d_v = 3$ et $d_c = 5$ où $a = 2$, $b = 5$.

$$H = \begin{bmatrix} I_1 & I_2 & I_4 & I_8 & I_{16} \\ I_5 & I_{10} & I_{20} & I_9 & I_{18} \\ I_{25} & I_{19} & I_7 & I_{14} & I_{28} \end{bmatrix} \tag{1.8}$$

Cette matrice de taille $N = 155$ est appelée *"matrice de Tanner"*. Elle a une distance minimale $d_{min} = 20$ et un diamètre de 6, lequel est le meilleur possible pour un code régulier $(3, 5)$ avec les paramètres $(N = 155,\ M = 93)$.

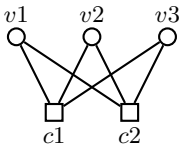
Récemment beaucoup de méthodes de conception de code QC-LDPC avec de large girth sont proposées. Ces méthodes sont basées soit sur des algorithmes de recherche pour trouver le décalage $p_{i,j}$ de chaque sous-matrice $I_{p_{i,j}}$ dans l'équation 1.5 [Fos04, O'S06, MKL06, WYD08, WYZ08, WDY13, ZW10, EG10, HHY12, KNCS07b] , soit elles sont explicites [ZZ14, ZSW12a, ZSW12b, ZSW13b, ZSW13a, LFK08, KNCS06, VPI04, AHK$^+$04, KPP$^+$04] . Une étude plus détaillée sur ces méthodes de construction de codes QC LDPC est faite dans le chapitre 2.

• Les Protographes

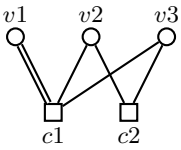
Un protographe binaire est défini comme un petit graphe biparti à partir duquel un graphe plus large est obtenu par une procédure de copies et permutation [Tho03]. Le nombre de copies dépend de la taille du code souhaité. Un exemple est représenté sur la figure 1.2. Le protographe est généralement décrit par la matrice d'adjacence associée appelée *matrice de base ou protomatrice* [LSL$^+$06, RU08, RDW15]. Elle est souvent notée H_B mais nous utilisons la notation H_P pour éviter des confusions avec la matrice des décalages des codes QC-LDPC. La matrice H_P est de dimension $M_p \times N_p$ et elle est remplie avec des valeurs entières. Ces valeurs représentent le nombre de branches entre le m^{eme} nœud de contrôle c_m et le n^{eme} nœud de variable v_n du protographe. Si les coefficients de H_P sont seulement des "0" et "1" le protographe est dit de *Type-I* (cf. figure 1.2(a)), s'il existe des coefficients supérieurs à "1" alors le protographe est de *Type-II* (cf. figure 1.2(b)) et si toutes les valeurs de H_P sont non nulles le protographe est dit *complet* [RDW15].

Exemple 1.5 *Les figures 1.2(a) et 1.2(b) sont respectivement un protographe type-I complet et un protographe type-II. Leurs matrices de base respectives sont :*

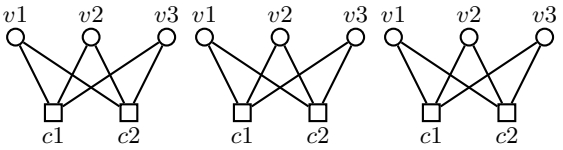
$$H_{P_1} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad H_{P_2} = \begin{bmatrix} 2 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$



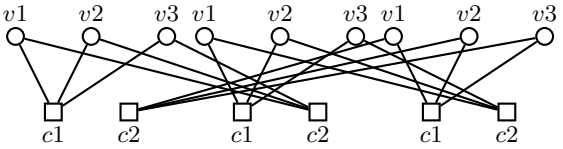
(a) Protographe Type-I



(b) Protographe Type-II



(c) trois copies du type-I



(d) Graphe final obtenu après permutation

FIGURE 1.2 – Procédure : Protographe, Copie et Permutation

La matrice de contrôle H est obtenue en remplaçant chaque valeur non nulle de H_P par une matrice circulante de dimension $N/N_p \times N/N_p$ et chaque "0" par la matrice nulle $N/N_p \times N/N_p$. Les décalages des matrices circulantes sont choisis tel que le girth du graphe final soit maximisé. Les protographes peuvent avoir des girths supérieurs à 12 [KNCS07b, PHNS10, PHNS13] mais les protographes complets de type-I qui sont des codes QC-LDPC ont des girths inférieurs à 12. Le problème principal sur la construction de codes basés sur les protographes est de trouver les matrices circulantes qui permettent de maximiser le girth. Nous proposons dans le chapitre 2 un algorithme qui permet de bien choisir les décalages comme mentionné sur la remarque 2.2.

1.3.2 Construction basée sur les géométries Finies

En plus des méthodes algébriques basées sur les matrices de permutation, nous avons des constructions basées sur les géométries finies qui sont introduites par Kou et al dans [KLF01]. Les codes obtenus avec les géométries finies, ne contiennent pas de cycle de longueur 4 [KLF01, VT01, JW02, KPP$^+$04]. De plus, ces codes ont de bonnes distances minimales et des performances à quelques dixièmes de decibels

dB de la limite théorique de Shannon [KLF01]. La construction de ces codes se base sur les points et lignes de la géométrie projective et de la géométrie euclidienne sur les corps finis. Nous avons deux types de code basés sur la géométrie euclidienne qui sont dénotés par Type-I EG-LDPC Codes et Type-II EG-LDPC Codes et deux autres types de code basés sur la géométrie projective qui sont dénoté par Type-I PG-LDPC Codes, Type-II PG-LDPC Codes. Ces types ne sont pas traités dans ce manuscrit. Mais il faut noter que toutes les géométries finies peuvent être mises en forme soit cyclique ou quasi-cyclique ; ce qui permet d'avoir un encodage en temps linéaire avec des registres de décalage et peu d'espace mémoire.

1.3.3 Construction combinatoire basée sur les BIBDs

Le graphe biparti est un outil trés utile pour la visualisation de l'algorithme de passage de messages, mais il n'est pas pratique dans la conception de code. Une solution pratique proposée est la construction combinatoire [JW01, VM04, AHK$^+$04, ZXMZ05, TKLK07, GH12]. Nous faisons une briève description de la conception combinatoire basée sur le *BIBD* pour Balanced Incomplete Block Design. Un *Balanced Incomplete Block Design* ou 2-designs est une paire $(C, \mathcal{B})$, où $C = \{c_0,c_1,\cdots,c_{M-1}\}$ est un ensemble de M éléments appelés *points*, $\mathcal{B}$= $\{B_0,B_1,\cdots,B_{N-1}\}$ est une collection de N sous-ensembles de C dont chacun à k points et est appelé *bloc* tel que :

- chaque élément de C soit contenu dans exactement r-blocs.
- chaque paire de points de C soit contenu dans exactement λ-blocs

Si tous les blocs contiennent le même nombre k de points et chaque point est contenu dans le même nombre de blocs r alors le BIBD est dit à *tactical configuration*. La notation $BIBD\,(M, k, \lambda)$ est utilisée pour désigner les BIBDs à M points avec des blocs de cardinal k et d'indice λ . Dans le cas où le cardinal des sous-ensembles vaut 3 ($k = 3$), le BIBD est appelé *Steiner triple system*. Si les blocs sont considérés comme les bits et les points comme les équations de parité alors la matrice de contrôle H appelée matrice incidente point-bloc de dimension $M \times N$. H est obtenue en considérant $h_{m,n} = 1$, si le $(m+1)^{eme}$ point est dans le $(n+1)^{eme}$ bloc c'est à dire $c_m \in B_n$.

Exemple 1.6 *: Soit un Steiner triple system tel que l'ensemble des points* $C = \{0,1,2,3,4,5,6\}$ *est associé à la collection des blocs* $\mathcal{B}$= $\{B_0,B_1,B_2,B_3,B_4,B_5,B_6\}$, *on peut définir :* $B_0 = \{0,1,2\}$, $B_1 = \{0,3,4\}$, $B_2 = \{0,5,6\}$, $B_3 = \{1,3,5\}$, $B_4 = \{1,4,6\}$, $B_5 = \{2,3,6\}$, $B_6 = \{2,4,5\}$. *La matrice incidente point-bloc correspon-*

dante est :

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \tag{1.9}$$

*Nous avons $\lambda = 1$ et $k = 3$ ce qui implique que $M = N$ donc $k = r$. Le BIBD est dit **symétrique**. Bien que la matrice soit carrée, elle définit un code car ces lignes sont dépendantes par exemple la somme des lignes l_0, l_1, l_4 et l_5 est nulle. En appliquant la relation définie par Vasic et al. dans [VM04], le code de l'exemple a un rendement $R \geq 1 - \frac{k(k-1)}{\lambda(M-1)} = 1/7$.*

Zhang et al, puis Tao et al. proposent dans [ZXMZ05, TKLK07] des constructions combinatoires de girth $g = 10$, et récemment des conceptions combinatoires basées sur les matrices circulantes sont développées [GH12, ZMZP08].

Nous pouvons remarquer que toutes les constructions structurées peuvent se baser sur des matrices circulantes. Et comme nous l'avons dit dans la présentation des codes QC-LDPC, cette structure de bloc-circulante conduit à des codeurs et décodeurs efficaces pour l'implémentation pratique. Cette efficacité s'explique par l'utilisation du parallélisme pour améliorer la vitesse d'encodage et le stockage des premières rangées de chaque sous-matrices pour réduire la mémoire du décodeur.

1.4 Constructions Pseudo-aléatoires

Les constructions pseudo-aléatoires donnent de bonnes performances pour les codes de longueurs importantes, en particulier pour les codes irréguliers, [Mac99, LMSS01, RSU00, RU01, RSU01, CFRU01, HEA01, HEA02, VL02, HEA05] où ces performances atteignent jusqu'à $0.0045dB$ de la limite de Shannon. Cependant, ces codes ne sont pas faciles à mettre en œuvre dans la pratique. De plus pour les codes de longueurs finies, la probabilité de choisir un graphe aléatoire défavorable est élevé [DPT$^+$02]. Une méthode très efficace pour construire un graphe de Tanner ayant une large girth est donné en utilisant l'algorithme PEG (Progressive Edge-Growth) [HEA01, HEA02, HEA05] que nous utilisons dans les chapitres 2 et 3. Nous faisons au préalable une description détaillée de cet algorithme dans cette section.

1.4.1 Algorithme Progressive Edge-Growth (PEG)

L'algorithme PEG est introduit par Hu et al [HEA01] et permet la création de graphe avec de larges girths. Pour un code de paramètres donnés :

- N, le nombre de nœuds de variable.
- M, le nombre de nœuds de contrôle.
- $D_v = \{d_{v_0}, d_{v_1}, \cdots, d_{v_{N-1}}\}$, l'ensemble des degrés des nœuds de variable où d_{v_n} est le degré du $(n+1)^{eme}$ nœud de variable

La construction du graphe de Tanner se fait de manière progressive ; nœud de variable par nœud de variable tout en maximisant le girth localement. Pour chaque nouveau nœud de variable v_n, la création d'une nouvelle branche $E^i_{v_n}$ se base sur l'expansion du sous graphe de v_n jusqu'à la profondeur k où le cardinal de l'ensemble des nœuds de contrôle présents sur le sous graphe n'évolue plus. Si à la profondeur k tous les nœuds de contrôle figurent dans le sous graphe étendu alors la nouvelle branche est créée par une connexion entre v_n et un nœud de contrôle c_m qui apparaît pour la première fois à cette profondeur k. Ainsi un ou plusieurs cycle(s) de longueur $2(k+1)$ sont créés (voir figure 1.3(a)). Tandis que, si tous les nœuds de contrôle ne sont pas présents dans le sous graphe étendu alors la nouvelle branche est construite entre v_n et un de ces nœuds non présents. Dans ce cas il n'y a pas de nouveau $2(k+1)$-cycle créé (voir figure 1.3(b)).

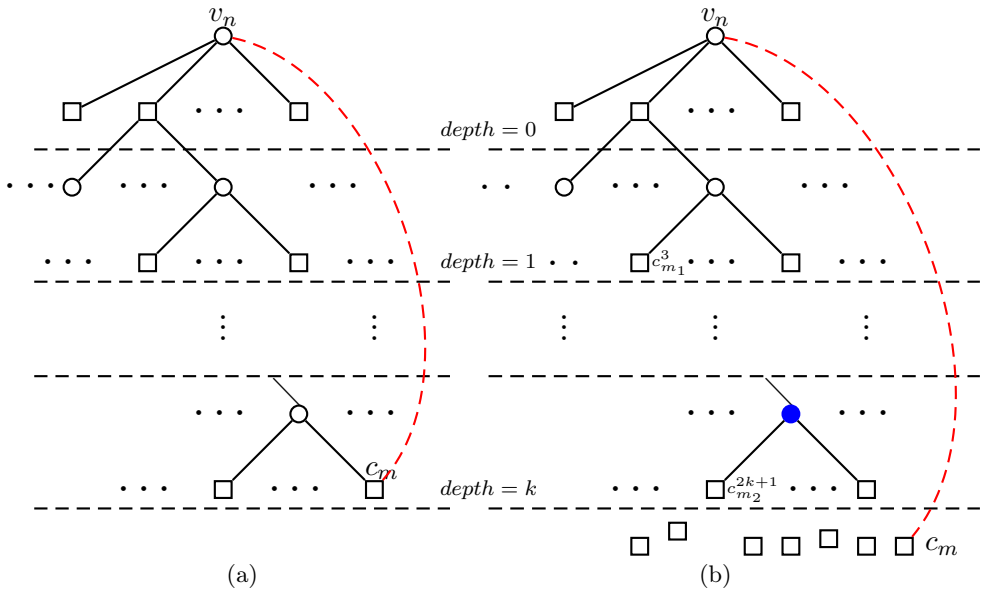


FIGURE 1.3 – Sous graphe expandu depuis le nœud de variable v_n, 1.3(a) : Nouvelle Branche en rouge créant un $2(k+1)$-cycle, 1.3(b) : Nouvelle Branche en verte ne créant pas de cycle

Le sous graphe étendu est appelé *arbre des voisins* ou encore *arbre de calcul* et est noté T_{v_n}. L'arbre a $2(k+1)$ niveaux indexés entre $[0, 2k+1]$, où le 0^{eme} niveau contient uniquement le nœud racine v_n, les niveaux pairs contiennent uniquement des nœuds de variable représentés par ○ tandis que les niveaux impairs contiennent uniquement des nœuds de contrôle représentés par □ (voir figure 1.3). L'ensemble des nœuds de contrôle présents aux niveaux $2k_i+1$ dans un arbre est noté par $\mathcal{M}_{v_n}^k$, où $k_i \leq k$. L'ensemble complémentaire à $\mathcal{M}_{v_n}^k$ est noté $\overline{\mathcal{M}}_{v_n}^k$. Ainsi l'algorithme original PEG peut être décrit comme suit :

Input : N, M, D_v
Output : G
for $n = 0$ **to** $N-1$ **do**
  for $i = 0$ **to** $d_{v_n} - 1$ **do**
    if *i=0* **then**
      $E_{v_n}^0 \leftarrow branche\,(c_m, v_n)$, où $E_{v_n}^0$ est la première branche incidente à v_n et c_m est le nœud de contrôle qui a le plus faible degré courant du sous graphe avec les branches existantes $E_{v_0} \cup E_{v_1} \cup E_{v_2} \cup \cdots E_{v_{n-1}}$.
    end
    else
      expandre, avec les branches existantes, le sous graphe depuis le nœud de variable v_n jusqu'à la profondeur k pour laquelle le cardinal de $\mathcal{M}_{v_n}^k$ ne s'augmente plus mais inférieur à M, ou bien $\overline{\mathcal{M}}_{v_n}^k \neq \emptyset$ mais $\overline{\mathcal{M}}_{v_n}^{k+1} = \emptyset$. $E_{v_n}^i \leftarrow branche\,(c_m, v_n)$, où $E_{v_n}^i$ est la i^{eme} branche incidente de v_n et c_m est un nœud de contrôle choisi aléatoirement dans l'ensemble des nœuds de contrôle appartenant à $\overline{\mathcal{M}}_{v_n}^k$ et ayant le plus faible degré.
    end
  end
end

Algorithme 1: Algorithm Original PEG

Cette version originale est appelé "*greedy*" du fait de l'expansion jusqu'à la profondeur k pour laquelle $\overline{\mathcal{M}}_{v_n}^k \neq \emptyset$ mais $\overline{\mathcal{M}}_{v_n}^{k+1} = \emptyset$. Hu et al proposent dans [HEA05] une version "*non-greedy*" qui consiste à tronquer l'arbre à la profondeur $k_{max} = g_t/2-2$, où g_t est le girth ciblé. Une des particularités de l'algorithme PEG est sa flexibilité. L'algorithme peut être utilisé pour construire des codes réguliers ($d_{v_n} = d_v$,) et des codes irréguliers avec des tailles arbitraires. Il y'a beaucoup de dérivées du PEG en particulier pour les codes irréguliers dont l'objectif principal est la gestion du profil d'irrégularité afin d'obtenir un code de large girth avec un faible plancher d'erreurs [HEA02, TJVW03, XB04, YB04, RH06, CC07, KKKS07, ZLT10]. Venkiah et al propose dans [VDP08] l'algorithme *RandPEG* qui permet

de construire un code de girth ciblé avec une réduction du nombre de cycles lors de la création d'une nouvelle branche. Dans la suite de ce manuscrit nous nous intéressons plus précisément à l'algorithme RandPEG pour la création de codes LDPC. Nous proposerons de nouveaux algorithmes qui généralisent le RandPEG. Ces nouveaux algorithmes intégrent des contraintes supplémentaires à celles du RandPEG et permettent d'optimiser deux problèmes. Le premier problème est la conception de code LDPC large girth avec des MPCs de taille faible. Le second problème est la supression et/ou la minimisation des trapping-sets/expansion-sets lors de l'élaboration des codes.

1.5 Conclusion

Dans ce chapitre nous avons présenté l'historique des codes LDPC, puis rappelé quelques définitions et notations, pour terminer par une bref description des méthodes de construction de codes LDPC. Au terme de ce tour d'horizon sur les méthodes de construction, nous pouvons constater que toutes ces méthodes ont pour objectif principal l'augmentation du girth et/ou la réduction de la complexité d'encodage et de décodage. Nous remarquons que toutes les constructions structurées (algébriques ou combinatoires) peuvent se convertir en code QC-LDPC qui sont favorables dans l'implémentation matérielle (encodage à temps linéaire et mémoire faible).

Chapitre 2

Construction de codes LDPC Quasi-Cycliques avec un large girth

Dans ce chapitre nous présentons nos deux premières contributions principales. La première contribution porte sur l'amélioration des bornes minimales des circulantes. Nous définissons une nouvelle borne qui est "tight" pour les codes QC-LDPC régulier $(3, d_c)$ de girth 8. La deuxième contribution est la conception d'un algorithme de construction RandPEG, utilisant le concept de *cycle non-vu*, et plus efficace que les algorithmes existants. Ce concept que nous avons introduit est applicable à toutes les valeurs de girth de g=8 à g=12. L'algorithme RandPEG construit efficacement des codes QC-LDPC en intégrant ce concept de cycle non-vu.

Ce chapitre est organisé comme suit : tout d'abord nous rappelons les généralités sur les codes quasi-cycliques en s'accentuant sur les cycles, ensuite nous présentons l'état de l'art sur les bornes minimales de la taille des MPCs, puis nous décrivons nos contributions novatrices sur les bornes minimales de la taille des MPCs. Après cette description nous proposons un algorithme RandPEG amélioré pour la conception de codes QC-LDPC, dans cette section nous donnons les conditions nécessaires et suffisantes pour éviter les cycles non-vus. Nous terminons par une étude comparative de performances de notre algorithme de construction avec les méthodes de constructions de codes QC-LDPC basées sur des algorithmes de recherches [Fos04, O'S06, WYD08, WDY13] ou sur des modèles explicites cycles [ZSW13b, ZZ14].

2.1 Généralités sur les codes Quasi-Cycliques

Les codes quasi-cycliques LDPC (QC-LDPC) sont une classe particulièrement importante des codes LDPC. Ces codes ont une matrice de contrôle composée de sous-matrices de permutation circulantes. Les structures circulantes permettent d'utiliser des parallélismes dans le décodeur ; ce qui est très important dans l'implémentation pratique. Ainsi les codes QC-LDPC sont présents dans une variété de normes de systèmes de communication, telles que IEEE 802.16e [802], DVB-S2 [DS] et 802.11.

Un code QC-LDPC régulier $\mathcal{C}(d_v, d_c)$ peut être représenté par sa matrice de parité qui est sous la forme :

$$H = \begin{bmatrix} I_0 & I_0 & \dots & I_0 & \dots & I_0 \\ I_0 & I_{p_{1,1}} & \dots & I_{p_{1,l}} & \dots & I_{p_{1,d_c-1}} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ I_0 & I_{p_{j,1}} & \dots & I_{p_{j,l}} & \dots & I_{p_{j,d_c-1}} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ I_0 & I_{p_{d_v-1,1}} & \dots & I_{p_{d_v-1,l}} & \dots & I_{p_{d_v-1,d_c-1}} \end{bmatrix} \tag{2.1}$$

où $1 \leq j \leq d_v - 1$, $1 \leq l \leq d_c - 1$ et $I_{p_{j,l}}$ représente la matrice de permutation circulante. Les codes QC-LDPC peuvent être également représentés par la *matrice de base ou matrice des décalages* qui est définie par :

$$H_B = \begin{bmatrix} 0 & 0 & \dots & 0 & \dots & 0 \\ 0 & p_{1,1} & \dots & p_{1,l} & \dots & p_{1,d_c-1} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & p_{j,1} & \dots & p_{j,l} & \dots & p_{j,d_c-1} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & p_{d_v-1,1} & \dots & p_{d_v-1,l} & \dots & p_{d_v-1,d_c-1} \end{bmatrix} \tag{2.2}$$

2.1.1 Caractérisation des cycles d'un code QC-LDPC

Pour comprendre comment est caractérisé un cycle dans la matrice de parité d'un code QC-LDPC nous utilisons l'approche de Fossorier [Fos04]. Un cycle de longueur $2i$ dans H est défini comme une suite de $2i$ positions telles que :

- deux positions consécutives sont obtenues par changement alternatif de ligne ou colonne seulement.
- Toutes les positions sont différentes à l'exception du premier et du dernier "1".

Donc deux positions consécutives dans un cycle appartiennent à des matrices de permutation circulantes distinctes qui sont soit dans le même bloc ligne ou soit dans le même bloc colonne. Ainsi un cycle de longueur $2i$ peut être associé à la série ordonnée de matrices de permutation circulantes :

$$I_{p_{j_0,l_0}}, I_{p_{j_1,l_0}}, I_{p_{j_1,l_1}}, \cdots, Ip_{j_{i-1},l_{i-1}}, I_{p_{j_0,l_{i-1}}}, I_{p_{j_0,l_0}}$$

avec $j_k \neq j_{k-1}$ et $l_k \neq l_{k-1}$, $\forall 1 \leq k \leq i$; où $j_0 \neq j_i$ et $l_0 \neq l_i$. Fossorier a énoncé dans [Fos04] qu'un cycle de longueur $2i$ existe si et seulement nous avons la relation suivante :

$$\sum_{k=0}^{i-1} \left(p_{j_k,l_k} - p_{j_{k+1},l_k}\right) = 0 \; mod \; p \tag{2.3}$$

Avec $j_k \neq j_{k+1}$ et $j_0 = j_i$ de même $l_k \neq l_{k+1}$ et $l_0 = l_i$. Cette équation caractérise un $2i$-cycle.

En effet, pour un cycle de longueur $2i$ nous avons $I(p_{j_0,l_0}) = I(p_{j_i,l_i})$. Soit r_k la ligne de $I(p_{j_k,l_k})$ dont la position à "1" est traversé par le cycle, donc nous avons :

$$r_0 = r_i \Rightarrow r_0 - r_i = 0,$$

en ajoutant et en retranchant r_k avec $1 \leq k \leq i-1$, on obtient :

$$r_0 - r_1 + r_1 - r_2 + r_2 - r_3 + \cdots\cdots - r_{i-2} + r_{i-2} - r_{i-1} + r_{i-1} - r_i = 0,$$

autrement,

$$\textstyle\sum_{k=0}^{i-1} (r_k - r_{k+1}) = 0$$

De plus, pour un cycle donné la ligne de $I_{p_{j_{k+1},l_k}}$ et la ligne de $I_{p_{j_{k+1},l_{k+1}}}$ traversées par ce cycle sont les mêmes et la colonne c_k traversée dans la circulante $I_{p_{j_k,l_k}}$ est la même que celle traversée dans $I_{p_{j_{k+1},l_k}}$. Par définition les lignes r_k, r_{k+1} et la colonne c_k vérifient les relations suivantes :

$$r_k = c_k + p_{j_k,l_k} \tag{2.4}$$

$$r_{k+1} = c_{k+1} + p_{j_{k+1},l_{k+1}} \tag{2.5}$$

$$= c_k + p_{j_{k+1},l_k} \tag{2.6}$$

les équations 2.4 et 2.6 impliquent

$$r_k - r_{k+1} = p_{j_k,l_k} - p_{j_{k+1},l_k}$$

Par suite, un cycle de longueur $2i$ existe si et seulement si

$$\sum_{k=0}^{i-1} \left(p_{j_k,l_k} - p_{j_{k+1},l_k}\right) = 0 \ \ mod\, p.$$

Remarque 2.1 *L'orientation des décalages (gauche ou droite) sur les matrices de permutation circulantes n'a pas d'impact sur les cycles. En effet, si les $p_{j,l}$ sont les décalages à droite alors pour chaque ligne r donnée, $0 \leq r \leq p-1$, nous avons un seul "1" à la colonne $c = r + p_{i,j}\, mod\, p$, ainsi*

$$c_k = r_k + p_{j_k,l_k} = r_{k+1} + p_{j_{k+1},l_k}\ mod\, p$$

et

$$r_k - r_{k+1} = p_{j_{k+1},l_k} - p_{j_k,l_k}\ mod\, p$$

Par suite, un cycle de longueur $2i$ existe si et seulement si :

$$\sum_{k=0}^{i-1} \left(p_{j_{k+1},l_k} - p_{j_k,l_k}\right) = 0 \ \ mod\, p \Rightarrow \sum_{k=0}^{i-1} \left(p_{j_k,l_k} - p_{j_{k+1},l_k}\right) = 0 \ \ mod\, p$$

L'équation 2.3 qui est uniquement en fonction des entrées de la matrice de base H_B, permet de définir le théorème suivant.

Théorème 2.1 *([Fos04]) Une condition nécessaire et suffisante pour qu'un graphe de Tanner, correspondant à la matrice de parité H définie par l'équation 2.1, ait un girth $g \geq 2i$ est :*

$$\sum_{k=0}^{q-2} \left(p_{j_k,l_k} - p_{j_{k+1},l_k}\right) \neq 0\ mod\, p,\ \forall 3 \leq q \leq i \tag{2.7}$$

Une conséquence de ce théorème 2.1 est le théorème suivant établi par Fossorier. Ce théorème donne une valeur maximale du girth pour les codes QC-LDPC régulier.

Théorème 2.2 *Pour tout code QC-LDPC régulier (d_v, d_c), le girth est inférieur ou égale à 12 ($g \leq 12$)*

En plus de ces théorèmes, Fossorier montre que pour chaque girth donné il existe une valeur minimale de la taille p des matrices de permutation circulantes.

2.2 Bornes minimales sur la taille des circulantes : état de l'art

Fossorier fut le premier à définir des valeurs minimales sur la taille p des matrices de permutation circulantes d'un code QC-LDPC régulier. Toutes les valeurs

minimales définies par Fossorier sont des conséquences du théorème 2.1. En effet, il donne des conditions nécessaires pour obtenir un code QC-LDPC de girth $2i$ avec i compris entre 3 et 6 à partir de l'équation 2.7. Ces conditions définissent les valeurs minimales de la taille des matrices de permutation circulantes pour chaque girth souhaité. Ces valeurs minimales sont aussi appelées bornes minimales et elles varient en fonction des paramètres d_v et d_c.

2.2.1 Code de Girth $g \geq 6$

D'après l'équation 2.7, la condition nécessaire pour avoir un code de girth $g \geq 6$ est :

$$\sum_{k=0}^{1} \left(p_{j_k,l_k} - p_{j_{k+1},l_k}\right) \neq 0 \; mod \; p, \tag{2.8}$$

Un résultat qui découle de cette équation est le théorème suivant élaboré par Fossorier :

Théorème 2.3 *Une condition nécessaire pour avoir un code de girth $g \geq 6$ est :*

$$\begin{array}{ll} p_{j_1,l_1} \neq p_{j_0,l_1} & \textit{pour } j_1 \neq j_0, \textit{ et} \\ p_{j_1,l_1} \neq p_{j_1,l_0} & \textit{pour } l_1 \neq l_0 \end{array}$$

Ce théorème montre que les entrées d'une ligne de H_B, exceptée la première, sont deux à deux différentes de même les entrées d'une colonne de H_B, exceptée la première, sont deux à deux différentes. Ainsi la valeur minimale de la taille des matrices de permutation circulantes pour un code de girth 6 est donnée par le corollaire suivant établi par Fossorier.

Corollaire 2.1 *Une condition nécessaire pour obtenir un code QC-LDPC de girth $g \geq 6$, est $p \geq d_c$.*

En plus de ce résultat général Fossorier donne, une condition nécessaire et suffisante suivant la parité de d_c. C'est le théorème 2.3 dans [Fos04] et il est énoncé comme suit :

Théorème 2.4 *Une condition nécessaire pour avoir $g \geq 6$ dans la représentation du graphe de Tanner d'un code QC-LDPC régulier (d_v, d_c) est $p \geq d_c$, ou $N \geq d_c^2$ si d_c est impair, et $p \geq d_c + 1$, ou $N \geq d_c(d_c + 1)$ si d_c est pair.*

Cette valeur minimale de p évolue en fonction du girth et est plus importante si $g > 6$.

2.2.2 Code de Girth $g \geq 8$

Toujours d'après l'équation 2.7, une condition nécessaire pour avoir un code de girth 8 est :

$$\sum_{k=0}^{q} \left(p_{j_k,l_k} - p_{j_{k+1},l_k}\right) \neq 0 \; mod \; p \;\; 1 \leq q \leq 2 \tag{2.9}$$

Avec la même approche que pour les codes de girth 6, Fossorier définit sur le théorème suivant une condition nécessaire entre les entrées de H_B pour obtenir un code de girth supérieur ou égale à 8.

Théorème 2.5 *Pour $d_v \geq 3$ et $d_c \geq 3$, une condition nécessaire pour avoir $g \geq 8$ est $p_{j_1,l_1} \neq p_{j_2,l_2}$ pour $0 < j_1 < j_2$ et $0 < l_1 < l_2$.*

Le cas $d_v = 2$ était déjà traité par le corollaire 2.1 dans [Fos04] où Fossorier avait montré que si $d_v = 2$ alors le girth est de la forme $4i$. Ce théorème 2.5 montre que les entrées qui ne sont pas sur la même ligne ou la même colonne sont deux à deux différentes. Ce résultat conduit au corollaire suivant qui donne la valeur minimale de la taille des circulantes pour un code de girth $g \geq 8$.

Corollaire 2.2 *Une condition nécessaire pour avoir $g \geq 8$ dans le graphe de Tanner d'un code QC-LDPC régulier (d_v, d_c) est $p > (d_v - 1)(d_c - 1)$ ou $N > (d_v - 1)(d_c - 1)d_c$.*

Sudarsan et al proposent dans [RDW15] une amélioration de cette borne minimale p pour les codes QC-LDPC de girth 8 avec $d_v = 3$. Cette nouvelle borne est donnée par le théorème suivant et est définie avec des conditions particulières.

Théorème 2.6 *Pour un code QC-LDPC régulier $(3, d_c)$ de girth $g \geq 8$, si toute différence de deux entrées d'une ligne j_0 est égale à la différence de deux entrées d'une autre ligne j_1 ou bien si toute différence de deux entrées d'une ligne j_0 est différente de la différence de deux entrées d'une autre ligne j_1 alors :*

$$p \geq 3\,(d_c - 1) - 1 \tag{2.10}$$

Autrement dit, si $\forall j_0 \neq j_1$ et $l_0 \neq l_1$ il existe $l_0^{'} \neq l_0$ et $l_1^{'} \neq l_1$ tel que $p_{j_0,l_0} - p_{j_0,l_0^{'}} \neq p_{j_1,l_1} - p_{j_1,l_1^{'}}$ ou bien si $\forall j_0 \neq j_1$ et $l_0 \neq l_1$ il existe $l_0^{'} \neq l_0$ et $l_1^{'} \neq l_1$ tel que $p_{j_0,l_0} - p_{j_0,l_0^{'}} = p_{j_1,l_1} - p_{j_1,l_1^{'}}$ alors $p \geq 3\,(d_c - 1) - 1$.

En plus de cette amélioration de la borne définie par Fossorier pour les codes de girth $g \geq 8$, de nouvelles bornes sont définies pour les codes de girth $g = 10$ et $g = 12$.

2.2.3 Code de Girth $g = 10$ et $g = 12$

Pour les codes de girth $g \geq 10$, les valeurs minimales sont analysées par Karimi et Banihashemi [KB13] et Kim et al [KCY13]. Tout d'abord Karimi et Banihashemi donnent dans [KB13] la condition nécessaire pour avoir un code QC-LDPC de girth $g \geq 10$ mais ils n'abordent pas de manière explicite le cas des codes de girth $g = 12$. Cette condition qui définie la borne minimale des codes de girth $g \geq 10$ est décrit par le théorème suivant :

Théorème 2.7 *Pour tout code QC-LDPC, la condition nécessaire pour avoir un girth $g \geq 10$ est :*

$$p \geq (d_v - 1)(d_c - 1) d_c + 1 \tag{2.11}$$

Pour le cas particulier où $d_v = 3$, Karimi et Banihashemi améliorent la borne avec le lemme suivant :

Lemme 2.1 *Pour tout code QC-LDPC régulier $(3, d_c)$, la condition nécessaire pour avoir un girth $g \geq 10$ est :*

$$p \geq 3(d_c - 1) d_c + 1 \tag{2.12}$$

Pour les codes QC-LDPC de girth $g = 12$, seul Kim et al. ont proposé une borne minimale dans [KCY13]. Ils définissent cette borne à travers le théorème suivant :

Théorème 2.8 *Soit H la matrice de contrôle d'un code QC-LDPC régulier (d_v, d_c). Si H a un girth $g = 12$ alors*

$$p \geq (d_v - 1)(d_c - 1) + (d_v - 1)^2 (d_c - 1)^2 + 1 \tag{2.13}$$

Avec ce dernier théorème nous disposons d'une borne minimale pour chaque girth d'un code QC-LDPC régulier (d_v, d_c). Cependant, les résultats pratiques obtenus avec les méthodes de construction existantes n'atteignent pas ces bornes minimales pour les codes de girth $g \geq 8$. Ainsi, dans le but d'évaluer l'optimalité des méthodes existantes, la question suivante se pose :"*est-ce que les bornes minimales proposées sont atteignables ?*". La réponse à cette question nécessite une nouvelle étude sur les bornes minimales.

2.3 Bornes minimales sur la taille des circulantes : contributions novatrices

Pour une nouvelle étude sur les bornes minimales des circulantes, nous utilisons principalement l'équation 2.7 du théorème 2.1 qui définit la condition nécessaire et suffisante pour obtenir un code QC-LDPC avec un girth souhaité.

2.3.1 Code de Girth $g \geq 6$

Un code est de girth $g \geq 6$ s'il ne contient pas de cycle de longueur 4 dans son graphe. La condition nécessaire et suffisante pour les entrées de H_B est alors :

$$\sum_{k=0}^{1} \left(p_{j_k,l_k} - p_{j_{k+1},l_k}\right) \neq 0 \; mod \; p, \tag{2.14}$$

Les différents scénarios de 4-cycle peuvent être représentés comme dans la figure 2.1 où les pointillés rouges indiquent le cycle de longueur 4 et les cercles pleins rouges indiquent les entrées non nulles. Nous supposons dans toute la suite que la nouvelle valeur est toujours celle correspondante à l'entrée sur le dernier bloc ligne et dernier bloc colonne. Pour éviter ces scénarios, chaque nouvelle valeur p_{j_1,l_1} de la matrice de base H_B doit satisfaire les contraintes définies par le corollaire suivant.

Corollaire 2.3 *Soit $p_{j,l}$ les entrées de la matrice de base H_B, pour avoir un code de girth $g \geq 6$, il faut pour tout $j_1 \neq 0$ et $l_1 \neq 0$:*

1. *$p_{j_1,l_1} \neq 0$ c'est à dire toutes les entrées de H_B qui ne sont pas sur la première ligne et/ou sur la première colonne ne sont pas nulles. Voir figure 2.1(a).*
2. *$p_{j_1,l_1} \neq p_{j_1,l_0}$ pour tout $0 \neq l_0 \neq l_1$ c'est à dire toutes les entrées de H_B d'une même ligne sont différentes exceptée la première ligne où tous les éléments valent "0". Voir figure 2.1(b).*
3. *$p_{j_1,l_1} \neq p_{j_0,l_1}$ pour tout $0 \neq j_0 \neq j_1$ c'est à dire toutes les entrées de H_B d'une même colonne sont différentes exceptée la première colonne où tous les éléments valent "0". Voir figure 2.1(c).*
4. *$p_{j_1,l_1} \neq p_{j_1,l_0} - p_{j_0,l_0} + p_{j_0,l_1}$ pour tout $l_0 \neq l_1$ et $j_0 \neq j_1$. Voir figure 2.1(d).*

Les contraintes (2) et (3) impliquent que $p \geq max\,(d_v, d_c)$ pour obtenir un code de girth $g \geq 6$; donc $p \geq d_c$ pour les codes QC-LDPC. Ce résultat est identique avec celui établi par Fossorier dans le corollaire 2.1.

Cette nouvelle approche pour trouver et prouver la borne minimale, basée sur des contraintes établies à partir des scénarios peut se généraliser aux cas plus

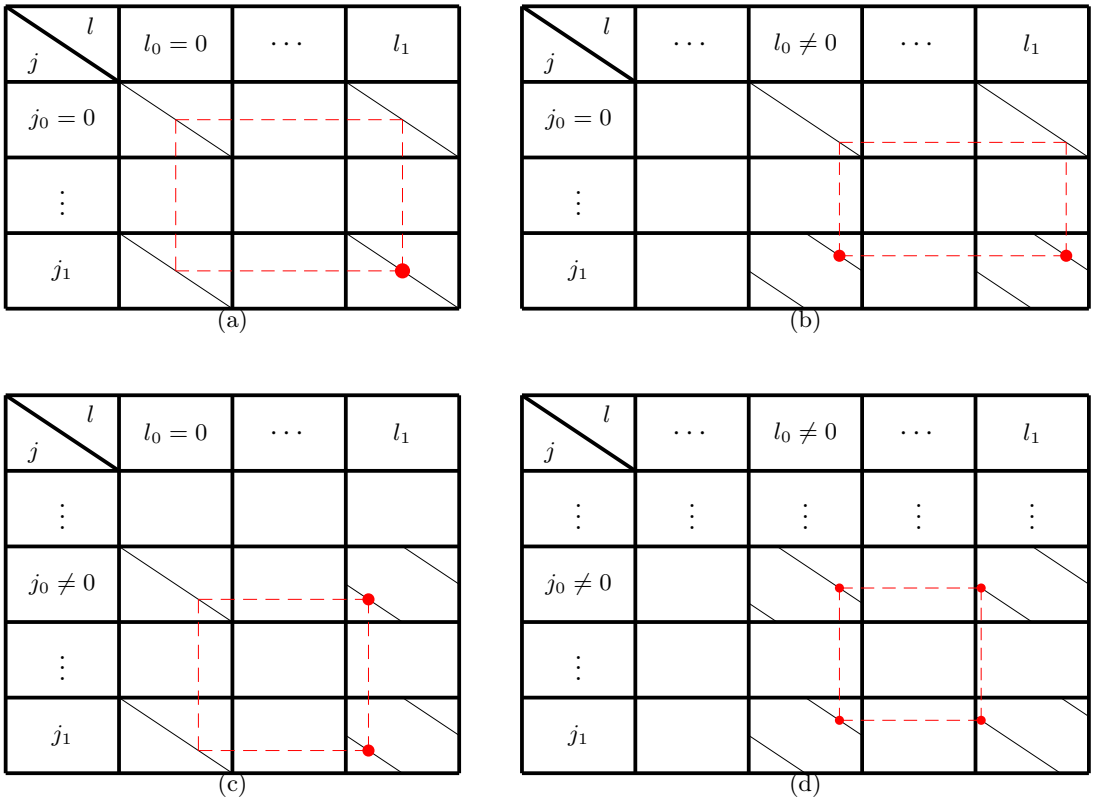


FIGURE 2.1 – Les Différents scénarios de 4-cycle

compliqués. Ainsi nous l'utilisons sur les codes de girth $g \geq 8$, vu qu'ils sont les codes où les résultats existants pour les valeurs minimales des circulantes sont différents des bornes minimales définies.

2.3.2 Code de girth $g \geq 8$

Un code est de girth 8 s'il ne contient pas de 4-cycle et de 6-cycle dans son graphe de Tanner. Cette condition nécessaire et suffisante est traduit par l'équation :

$$\sum_{k=0}^{q} \left(p_{j_k,l_k} - p_{j_{k+1},l_k}\right) \neq 0 \; mod \; p \;\; 1 \leq q \leq 2 \tag{2.15}$$

Le cas $q = 1$ est déjà traité sur les codes de girth $g \geq 6$. Le cas $q = 2$ traduit que le graphe de H ne contient pas de 6-cycle. Supposons que le graphe de H contient des 6-cycles alors nous avons :

$$\textstyle\sum_{k=0}^{2} \left(p_{j_k,l_k} - p_{j_{k+1},l_k}\right) = 0 \; mod \; p$$

avec $j_0 = j_3$, $j_k \neq j_{k+1}$, et $l_k \neq l_{k+1}$.

Les relations $j_0 \neq j_1 \neq j_2 \neq j_0$ et $l_0 \neq l_1 \neq l_2 \neq l_0$ impliquent qu'un 6-cycle est composé toujours de trois *bloc-lignes* différents et de trois *bloc-colonnes* différents. Ainsi pour $d_v = 3$, tous les 6-cycles traversent le première bloc-ligne dont les valeurs de décalage sont toutes nulles ($p_{0,l} = 0$). Cependant, tous les 6-cycles formés ne traversent pas obligatoirement la première colonne. Les différents scénarios d'un 6-cycle peuvent être représentés comme sur la figure 2.2 où les pointillés rouges indiquent le 6-cycle et les cercles pleins rouges indiquent les entrées non nulles.

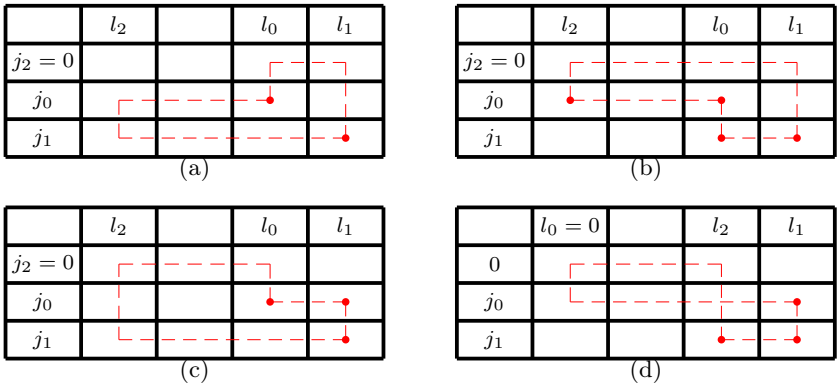


FIGURE 2.2 – Les différents scénarios de 6-cycle

Corollaire 2.4 *Soit $p_{j,l}$ les entrées de la matrice de base H_B. Pour avoir un code de girth $g \geq 8$, il faut pour tout $j_1 \neq 0$ et $l_1 \neq 0$:*

1. $p_{j_1,l_1} \neq p_{j_0,l_0}$, *pour tout* $0 \neq j_0 \neq j_1 \neq 0$, *tout* $0 \neq l_0 \neq l_1 \neq 0$ *et* $l_2 = 0$. *Toutes les entrées de H_B qui ne sont pas sur la même ligne et sur la même colonne sont différentes. Voir figure 2.2(a)*
2. $p_{j_1,l_1} \neq p_{j_1,l_0} - p_{j_0,l_0}$, *pour tout* $0 \neq j_0 \neq j_1 \neq 0$, *tout* $0 \neq l_0 \neq l_1 \neq 0$ *et* $l_2 = 0$. *Voir figure 2.2(b).*
3. $p_{j_1,l_1} \neq p_{j_1,l_0} - p_{j_0,l_0} + p_{j_0,l_2}$, *pour tout* $0 \neq j_0 \neq j_1$, *tout* $l_1 \neq l_0 \neq l_2 \neq l_1$ *et* $l_{k_{k\in\{0,1,2\}}} k \neq 0$. *Voir figure 2.2(b).*
4. $p_{j_1,l_1} \neq p_{j_0,l_1} - p_{j_0,l_0}$, *pour tout* $0 \neq j_0 \neq j_1 \neq 0$, *tout* $0 \neq l_0 \neq l_1 \neq 0$, $l_2 = 0$ *Voir figure 2.2(c).*
5. $p_{j_1,l_1} \neq p_{j_0,l_1} + p_{j_1,l_2}$, *pour tout* $0 \neq j_0 \neq j_1 \neq 0$, *tout* $0 \neq l_2 \neq l_1 \neq 0$, $l_0 = 0$. *Voir figure 2.2(d).*
6. $p_{j_1,l_1} \neq p_{j_0,l_1} - p_{j_0,l_0} + p_{j_1,l_2}$, *pour tout* $0 \neq j_0 \neq j_1$, *tout* $l_1 \neq l_0 \neq l_2 \neq l_1$ *et* $l_{k_{k\in\{0,1,2\}}} k \neq 0$. *Voir figure 2.2(c).*

La première contrainte du corollaire 2.4 et les trois premières conditions du corollaire 2.3 impliquent que *toutes les entrées de la matrice de base H_B sont différentes deux à deux, à l'exception de la première ligne et de la première colonne où les entrées valent "0"*. Par conséquent, pour avoir un code QC-LDPC de girth $g \geq 8$ il faut que $p \geq (d_v - 1)(d_c - 1) + 1$. Ce résultat est identique à celui donné par Fossorier dans le corollaire 2.2 mais il ne tient pas compte de toutes les contraintes pour éviter un 6-cycle. Cette borne peut être améliorée davantage avec une hypothèse plus simple que celle proposée par Sudarsan et al. dans le théorème 2.6. La nouvelle borne minimale est donnée par le théorème suivant :

Théorème 2.9 *Pour un code QC-LDPC régulier $(3, d_c)$ de girth $g \geq 8$, s'il existe une ligne j, $1 \leq j \leq 2$ telle que toute différence de deux entrées sur cette ligne soit égale à une autre entrée de cette même ligne alors :*

$$p \geq 3(d_c - 1) \tag{2.16}$$

Autrement dit, si $\forall\ 1 \leq l_{i_{i=1,2}} \leq L-1,\ \exists\ l$ *tel que* $p_{j,l_1} - p_{j,l_2} = p_{j,l}$ alors $p \geq 3(d_c - 1)$

Preuve : *Regroupons toutes les contraintes des corollaires 2.3 et 2.4. La condition nécessaire pour avoir un code de girth $g \geq 8$ peut se résumer à :*

$$p_{j_1,l_1} \neq \begin{cases} 0 & \\ p_{j_0,l_1} & \\ p_{j_1,l_k} & 4-cycle \\ p_{j_1,l_k} - p_{j_0,l_k} + p_{j_0,l_1} & \\ \hline p_{j_0,l_k} & \\ p_{j_1,l_k} - p_{j_0,l_k} & \\ p_{j_1,l_{k_1}} - p_{j_0,l_{k_1}} + p_{j_0,l_{k_2}} & \\ p_{j_0,l_1} - p_{j_0,l_k} & 6-cycle \\ p_{j_0,l_1} + p_{j_1,l_k} & \\ p_{j_0,l_1} - p_{j_0,l_{k_1}} + p_{j_1,l_{k_2}} & \end{cases} \tag{2.17}$$

pour tout p_{j_1,l_1} avec $0 \neq j_0 \neq j_1 \neq 0$ et $0 \neq l_1 \neq l_k \neq 0$. Sans perdre la généralité, prenons $l_1 = d_c - 1$, alors les contraintes de l'équation 2.17 donnent :

• *pour* $j_1 = 1$ *et* $j_0 = 2$	• *pour* $j_1 = 2$ *et* $j_0 = 1$
$p_{1,d_c-1} \neq \begin{cases} 0 \\ p_{2,d_c-1} \\ p_{1,l} \\ p_{1,l} - p_{2,l} + p_{2,d_c-1} \\ p_{2,l} \\ p_{1,l} - p_{2,l} \\ p_{1,l_{k_1}} - p_{2,l_{k_1}} + p_{2,l_{k_2}} \\ p_{2,d_c-1} - p_{2,l} \\ p_{2,d_c-1} + p_{1,l} \\ p_{2,d_c-1} - p_{2,l_{k_1}} + p_{1,l_{k_2}} \end{cases}$	$p_{2,d_c-1} \neq \begin{cases} 0 \\ p_{1,d_c-1} \\ p_{2,l} \\ p_{2,l} - p_{1,l} + p_{1,d_c-1} \\ p_{1,l} \\ p_{2,l} - p_{1,l} \\ p_{2,l_{k_1}} - p_{1,l_{k_1}} + p_{1,l_{k_2}} \\ p_{1,d_c-1} - p_{1,l} \\ p_{1,d_c-1} + p_{2,l} \\ p_{1,d_c-1} - p_{1,l_{k_1}} + p_{2,l_{k_2}} \end{cases}$

En éliminant les contraintes identiques, on obtient :

$p_{1,d_c-1} \neq \begin{cases} 0 \\ p_{2,d_c-1} \\ p_{1,l} \\ p_{1,l} - p_{2,l} + p_{2,d_c-1} \\ p_{2,l} \\ p_{1,l} - p_{2,l} \\ p_{1,l_{k_1}} - p_{2,l_{k_1}} + p_{2,l_{k_2}} \\ p_{2,d_c-1} - p_{2,l} \\ p_{2,d_c-1} - p_{2,l_{k_1}} + p_{1,l_{k_2}} \end{cases}$	$p_{2,d_c-1} \neq \begin{cases} p_{2,l} \\ p_{1,l} \\ p_{1,d_c-1} - p_{1,l} \\ p_{2,l} - p_{1,l} \\ p_{2,l_{k_1}} - p_{1,l_{k_1}} + p_{1,l_{k_2}} \end{cases}$

Soient les ensembles définis comme suit :

- $S_0 = \{0\}$
- $S_1 = \{p_{1,l}\}$
- $S_1' = \{p_{1,d_c-1}\}$
- $S_2 = \{p_{2,l}\}$
- $S_2' = \{p_{2,d_c-1}\}$
- $S_3 = \{p_{2,d_c-1} + p_{1,l} - p_{2,l}\}$
- $S_4 = \{p_{2,d_c-1} - p_{2,l}\}$
- $S_5 = \{p_{1,d_c-1} - p_{1,l}\}$
- $S_6 = \{p_{1,l} - p_{2,l}\}$
- $S_7 = \{p_{2,l} - p_{1,l}\}$
- $S_8 = \left\{p_{2,d_c-1} - p_{2,l_{k_1}} + p_{1,l_{k_2}}\right\}$
- $S_9 = \left\{p_{1,l_{k_1}} - p_{2,l_{k_1}} + p_{2,l_{k_2}}\right\}$
- $S_{10} = \left\{p_{2,l_{k_1}} - p_{1,l_{k_1}} + p_{1,l_{k_2}}\right\}$

Le facteur de décalage p satisfait à la relation $p \geq min\left|\bigcup_{n=0}^{10} S_n\right|$. Pour un code de girth 8, nous avons :

- *Toutes les entrées de la matrice de base H_B sont différentes deux à deux donc $\bigcap_{n=0}^{2} S_n \bigcap S_1' \bigcap S_2' = \varnothing$ et $\left|\bigcup_{n=0}^{2} S_n \bigcup S_1' \bigcup S_2'\right| = 2(d_c - 2) + 1 + 1 + 1 = 2(d_c - 1) + 1$ (Voir Annexes B et C).*

- $S_3 \bigcap \left\{ S_0 \bigcup S_1 \bigcup S_1' \bigcup S_2' \right\} = \varnothing$ *mais* $S_3 \bigcap S_2$ *peut être non vide.*
- $S_4 \bigcap S_3 = \varnothing$. *Si* $\forall\ 1 \leq l_1, l_2 \leq d_c - 1,\ \exists\ l$ *tel que* $p_{2,l_1} - p_{2,l_2} = p_{2,l}$ *alors* $S_4 = S_2$ *or* $S_4 \bigcap S_3 = \varnothing$ *donc* $S_2 \bigcap S_3 = \varnothing$ *par conséquent* $S_3 \bigcap \left\{ S_0 \bigcup S_1 \bigcup S_2 \bigcup S_1' \bigcup S_2' \right\} = \varnothing$ *et* $\left| S_0 \bigcup S_1 \bigcup S_2 \bigcup S_1' \bigcup S_2' \bigcup S_3 \right| = 2(d_c - 1) + 1 + d_c - 2 = 3(d_c - 1)$.

Comme nous l'avons dit dans la preuve, le facteur de décalage p doit satisfaire à la relation $p \geq min \left| \bigcup_{n=0}^{10} S_n \right|$. La borne minimale atteignable est alors le cardinal de l'union des ensembles $S_{n_{n\leq 10}}$ $\left(\left| \bigcup_{n=0}^{10} S_n \right| \right)$. Or les ensembles $S_{n_{n\leq 10}}$ ne sont pas tous disjoints deux à deux, ce qui rend très complexe la détermination du cardinal $\left| \bigcup_{n=0}^{10} S_n \right|$. Par des tests numériques, nous avons évalué $\left| \bigcup_{n=0}^{10} S_n \right|$ et son évolution conduit à la conjecture suivante :

Conjecture 2.1 *Une condition nécessaire pour obtenir un girth* $g \geq 8$ *dans le graphe de Tanner d'un code QC-LDPC régulier* $(3, d_c)$ *est :*

$$p \geq 4\,(d_c - 2) + 1 \tag{2.18}$$

Il faut noter que cette borne peut être définie avec certaines hypothèses. En effet, pour chaque l_k^* fixé pour S_8 et S_{10} tel que :

$S_8^* = \left\{ p_{2,d_c-1} - p_{2,l_{k_1}} + p_{1,l_k^*} \right\}$ et $S_{10}^* = \left\{ p_{2,l_{k_1}} - p_{1,l_{k_1}} + p_{1,l_k^*} \right\}$

nous avons

$|S_8^*| = |S_{10}^*| = d_c - 3$ et $S_8^* \bigcap S_{10}^* = \varnothing$ si le girth est 8.

Si nous supposons qu'il existe un l_k^* tel que :

$$\left\{ \bigcup_{n=0}^{2} S_n \bigcup S_1' \bigcup S_2' \right\} \bigcap S_8^* = \varnothing \text{ et } \left\{ \bigcup_{n=0}^{2} S_n \bigcup S_1' \bigcup S_2' \right\} \bigcap S_{10}^* = \varnothing$$

alors

$$\begin{aligned} \left| \bigcup_{n=0}^{2} S_n \bigcup S_1' \bigcup S_2' \bigcup S_8^* \bigcup S_{10}^* \right| &= 2(d_c - 1) + 1 + 2(d_c - 3) \\ &= 4(d_c - 2) + 1. \end{aligned}$$

Cette approche de détermination de la borne minimale avec le cardinal des ensembles S_n peut être généralisée pour les codes de girth 10 et 12. Pour les bornes minimales des codes de girth 10, il suffit de déterminer les différents scénarios qui caractérisent les 8-cycles et de définir des contraintes à partir de ces nouveaux scénarios puis de combiner ces contraintes avec celles de l'équation 2.17. Il faut procéder de la même façon avec les 10-cycles pour avoir les bornes minimales des codes de girth 12. Nous n'avons pas eu le temps de traiter ces 2 cas (g=10 et g=12), mais c'est un travail en cours.

2.4 Conception de Codes QC-LDPC avec l'algorithme RandPEG

Ces dernières années, plusieurs travaux ont été effectués dans le but d'avoir des codes QC-LDPC de girth $g \geq 8$ avec de bon rendement, c'est à dire des code QC-LDPC de girth $g \geq 8$ avec un facteur de décalage $p \approx p_{min}$ où p_{min} est la borne minimale nécessaire. Parmi les méthodes de constructions proposées, nous avons des méthodes explicites dont la meilleure borne minimale est $d_c\left(d_c + [d_c\,(mod\,2)]\right)/2 + 1$, pour un code QC-LDPC régulier $(3, d_c)$ de girth 8 [ZSW13b]. Nous avons aussi des méthodes basées sur des algorithmes de recherche dont l'algorithme "*Hill-Climbing*" [WYD08] présente les meilleures bornes dans la littérature. Nous décrirons son principe de fonctionnement dans la section 2.6.

L'algorithme PEG, décrit dans la construction aléatoire 1.4.1, est très bien connu pour la construction de codes LDPC à large girth. Dans cette thèse nous nous intéressons à une de ses versions améliorées : l'algorithme *RandPEG*. Dans cette section nous rappelons le principe de fonctionnement de cet algorithme et nous décrivons ses anomalies pour des codes de girth $g \geq 8$. Puis nous donnons des solutions pour éviter les anomalies, ensuite nous proposons un algorithme RandPEG modifié qui représente une des contributions principales dans cette thèse. Enfin, nous faisons une comparaison de performances avec les méthodes de construction existant.

2.4.1 Algorithme RandPEG et cycles non-vus

L'algorithme *RandPEG* est introduit par Venkiah et al [VDP08]. Cet algorithme est amélioration du PEG pour construire des codes de girth ciblé. Cette amélioration consiste :

- à la troncature de l'arbre de calcul en fonction du girth ciblé g_t, $depth_{max} = g_t/2 - 1$
- à l'ajout d'une fonction coût qui évalue le nombre d'occurences de chaque nœud de contrôle c_m dans l'arbre
- à choisir aléatoirement un nœud de contrôle parmi les candidats ayant la fonction de coût minimum.

Le nombre d'occurences d'un nœud de contrôle c_m traduit le nombre de nouveaux cycles créé par une connexion entre v_n et c_m. Donc le RandPEG permet la construction de codes LDPC avec un girth ciblé en minimisant le nombre de nouveaux cycles locals. Nous utilisons les mêmes notations que celles dans l'al-

gorithme PEG de la partie 1.4.1, l'algorithme RandPEG est décrit comme suit :

Input : N, M, D_v, g_t
Output : G
for $n = 0$ **to** $N-1$ **do**
  for $i = 0$ **to** $d_{v_n} - 1$ **do**
    if *i=0* **then**
      $E^0_{v_n} \leftarrow branche\,(c_m, v_n)$, où $E^0_{v_n}$ est la première branche incidendente à v_n et c_m est le nœud de contrôle qui a le plus faible degré courant du sous graphe avec les branches existantes $E_{v_0} \cup E_{v_1} \cup E_{v_2} \cup \cdots E_{v_{n-1}}$.
    end
    else
      étendre, avec les branches existantes, le sous graphe depuis le nœud de variable v_n jusqu'à la profondeur $k = g_t/2 - 1$
      if $\overline{\mathcal{M}}^k_{v_n} \neq \varnothing$ **then**
        Choix aléatoire d'un nœud de contrôle c_m dans $\overline{\mathcal{M}}^k_{v_n}$:
        $E^i_{v_n} \leftarrow branche\,(c_m, v_n)$ (pas de cycle de longueur g_t créé)
      end
      if $\overline{\mathcal{M}}^k_{v_n} = \varnothing$ **then**
        if $\mathcal{M}^k_{v_n} \neq \varnothing$ **then**
          Etape 1 : Elimine les nœuds de profondeur inférieure à k
          Etape 2 : Elimine tous les nœuds de contrôle qui ont un nombre d'occurence $nbCycles_{c_m}$ plus grand que la valeur minimale des occurences $min\,(nbCycles_{c_m})$
          Etape 3 : Mettre à jour $\mathcal{M}^k_{v_n}$
        end
        if $\mathcal{M}^k_{v_n} \neq \varnothing$ **then**
          Choix aléatoire d'un nœud de contrôle c_m dans $\mathcal{M}^k_{v_n}$:
          $E^i_{v_n} \leftarrow branche\,(c_m, v_n)$
        end
        else
          Echec de l'algorithme
        end
      end
    end
  end
end

Algorithme 2: Algorithme RandPEG

L'algorithme RandPEG, comme tous les algorithmes dérivés du PEG, s'adapte facilement sur les codes QC-LDPC. En effet pour les codes QC-LDPC nous avons uniquement besoin des entrées de H_B ; c'est à dire les connexions des nœuds de variable $v_n = l.p$ avec $0 \leq l \leq d_c - 1$. Les connexions pour les nœuds restants v_n avec $l.p < v_n < (l+1).p$, seront déduits avec les structures circulantes. Sur l'algorithme PEG et ses dérivées, il suffit d'appliquer un "*pas*" de p sur la boucle des nœuds de variable.

Le saut de pas p implique le non traitement des cycles formés par les nœuds de variable compris entre $v_{l.p} + 1$ et $v_{(l+1).p} - 1$. Ainsi un cycle qui contient le nœud $v_{l.p}$ et v_n dans la même circulante, avec $l.p < v_n < (l+1)\,p$, n'est pas détecté

lors de l'expansion de l'arbre de racine $v_{l.p}$ car le nœud v_n n'existe pas encore à ce niveau de la construction. Ce résultat entraine que l'algorithme PEG et ses dérivées sont des algorithmes non-prédictifs pour la construction des codes QC-LDPC. Donc pour respecter et garantir les prédictions définies, une vérification à postériori est nécessaire. Cette vérification couteuse est un problème d'autant plus important que la taille des circulantes p augmente. D'où le besoin d'identifier de manière prédictive les noeuds de contrôle candidats dont leurs sélections génèrent des cycles qui n'étaient pas détectés lors de l'expansion de l'arbre.

Définition 2.1 *Le cycle est dit* **non-vu**, *si après élaboration de la matrice circulante dont le décalage est obtenu à partir du PEG, ce cycle qui ne figure pas sur l'arbre est créé.*

Caractérisation 2.1 *Tout cycle qui contient le nœud de variable $v_{l.p}$ et qui traverse au moins deux fois le nouveau bloc de matrice $I\left(p_{j,l}\right)$ est un cycle non-vu.*

D'après cette caractérisation, nous déterminons quels sont les cycles qui peuvent avoir au moins un bloc de matrice traversés plusieurs fois.

D'après l'équation 2.3, un 4-cycle est donné par la relation :

$$\sum_{k=0}^{1}\left(p_{j_k,l_k}-p_{j_{k+1},l_k}\right)=0\ mod\ p, \text{ où } j_0\neq j_1 \text{ et } l_0\neq l_1$$

Avec $j_0 \neq j_1$ et $l_0 \neq l_1$, les 4-cycles sont formés par deux bloc lignes et deux bloc colonnes. Par conséquent aucun des blocs qui constituent un 4-cycle ne peut être traversé deux fois.

De même pour les 6-cycles où nous avons :

$$\sum_{k=0}^{2}\left(p_{j_k,l_k}-p_{j_{k+1},l_k}\right)=0\ mod\ p$$
$$\text{où } j_0\neq j_1\neq j_2 \text{ et } l_0\neq l_1\neq l_2$$

Avec $j_0 \neq j_1 \neq j_2$ et $l_0 \neq l_1 \neq l_2$ les 6-cycles sont formés par trois bloc lignes et trois bloc colonnes. Par conséquent aucun des blocs qui constituent un 6-cycle ne peut être traversé deux fois. Par contre, pour les 8-cycles l'équation 2.3 donne :

$$\sum_{k=0}^{3}\left(p_{j_k,l_k}-p_{j_{k+1},l_k}\right)=0\ mod\ p$$
$$\text{où } j_0\neq j_1\neq j_2\neq j_3\neq j_0 \text{ et } l_0\neq l_1\neq l_2\neq l_3\neq l_0$$

Avec $j_0 \neq j_1 \neq j_2 \neq j_3 \neq j_0$ et $l_0 \neq l_1 \neq l_2 \neq l_3 \neq l_0$, nous pouvons avoir $j_0 = j_2,\ j_1 = j_3,\ l_0 = l_2$, ou encore $l_1 = l_3$. Ces quatre cas font qu'un bloc correspondant à la matrice $I\left(p_{j,l}\right)$ peut être traversé deux fois par un 8-cycle d'où la possibilité d'avoir un cycle non-vu avec l'algorithme PEG et ses dérivées pour un code QC-LDPC de girth $g \geq 8$. Par conséquent l'algorithme PEG et ses dérivées ne permettent plus de construire de manière prédictive des codes de girth $g \geq 10$.

Exemple 2.1 *On désire obtenir un code de girth $g = 10$, pour le code QC-LDPC régulier (3,4) avec $N = 37 \times 4$. Avec l'algorithme RandPEG, nous obtenons la matrice H suivante :*

$$H = \begin{bmatrix} I(0) & I(0) & I(0) & I(0) \\ I(0) & I(29) & I(35) & I(7) \\ I(0) & I(16) & I(36) & I(26) \end{bmatrix}$$

Pour $v_{2\times 37} = v_{74}$, après expansion de l'arbre, nous avons deux nœuds de contrôle c_{99} et c_{110} comme les potentiels candidats et le choix s'est porté sur c_{110}. Or cette nouvelle branche $E^k_{v_{74}} \leftarrow branche(c_{110}, v_{74})$ donne deux cycles de longueur 8 qui ne respectent pas le girth ciblé. Ces deux cycles sont formés des nœuds et blocs suivants :

- *Les nœuds de variable du premier 8-cycle sont $\{0, 36, 74, 75\}$ et les nœuds de contrôle sont $\{0, 73, 74, 110\}$. La série ordonnée de matrices de permutation circulantes associée au cycle est : $I(0)$, $I(0)$, $I(36)$, $I(35)$, $I(0)$, $I(0)$, $I(36)$, $I(0)$, $I(0)$ où le $1^{er} I(0)$ est la circulante au 1^{er} bloc ligne et 1^{er} bloc colonne, le $2^{eme} I(0)$ est la circulante au 3^{eme} bloc ligne et 1^{er} bloc colonne, le $3^{eme} I(0)$ est la circulante au 2^{eme} bloc ligne et 1^{er} bloc colonne, le $4^{eme} I(0)$ est au même bloc que 2^{eme}, le $5^{eme} I(0)$ est la circulante au 1^{er} bloc ligne et 3^{eme} bloc colonne et le dernier $I(0)$ est le 1^{er} $I(0)$ par définition d'un cycle. Le cycle traverse deux fois le bloc de matrice du $2^{eme} I(0)$ avec les nœuds v_0, v_{36} et c_{74}, c_{110} ; le cycle traverse aussi deux fois le nouveau bloc de matrice $I(36)$ avec les nœuds v_{74}, v_{75} et c_{74}, c_{110}.*
- *Les nœuds de variable du deuxième 8-cycle sont $\{35, 36, 74, 110\}$ et les nœuds de contrôle sont $\{36, 72, 109, 110\}$. C'est la même série ordonnée de matrices de permutation circulantes que celle du cycle précédent donc nous avons les mêmes interprétations. Le cycle traverse deux fois le nouveau bloc de matrice $I(36)$ avec v_{74}, v_{110} et c_{109}, c_{110}.*

Cependant, avec le nœud c_{99} et la branche (c_{99}, v_{74}), le girth local serait bien $g = 10$.

Nous proposons des conditions nécessaires et suffisantes pour éviter les cycles non-vus de longueur 8 et 10 ; nous permettant ainsi de construire des codes QC-LDPC de girth respectives g=10 et g=12.

2.4.2 Caractérisation des 8-cycles non-vus

Comme nous l'avons explicité ci-dessus, les 8-cycles satisfont à l'équation :

$$\sum_{k=0}^{3} (p_{j_k,l_k} - p_{j_{k+1},l_k}) = 0 \; mod \; p \tag{2.19}$$

avec $j_0 \neq j_1 \neq j_2 \neq j_3 \neq j_0$, $l_0 \neq l_1 \neq l_2 \neq l_3 \neq l_0$ mais nous pouvons avoir $j_0 = j_2$, $j_1 = j_3$, $l_0 = l_2$, ou encore $l_1 = l_3$. Dans tous les scénarios que nous traitons pour éviter les 8-cycles non-vus, nous supposons que le *nouveau bloc en vert est traversé deux fois et qu'il est celui formé par le dernier bloc ligne et le dernier bloc.* L'indice de ce nouveau bloc est (j_3, l_3) avec $j_3 = j_1$, $l_3 = l_1$. Nous notons $p^{\alpha}_{j,l}$ le bloc qui représente la sous matrice $I(p_{j,l})$ dans le cycle, avec α est le nombre de fois que le cycle traverse $I(p_{j,l})$.

• **8-Cycle non-vu formé par 2×2 blocs**

Soit un 8-cycle tel que tous les blocs qui le forment soient traversés deux fois. Donc $j_0 = j_2$ et $l_0 = l_2$ comme le montre la figure 2.3.

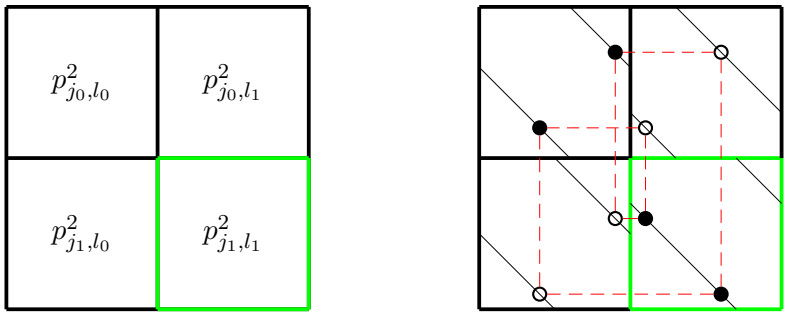


FIGURE 2.3 – 8-cycle formé par 2×2 blocs

L'équation 2.19 devient :

$$2(p_{j_0,l_0} - p_{j_1,l_0}) + 2(p_{j_1,l_1} - p_{j_0,l_1}) = 0 \; mod \; p \tag{2.20}$$

Cette équation conduit au théorème suivant.

Théorème 2.10 *La condition nécessaire et suffisante pour éviter un 8-cycle non-vu formé par 2×2 blocs est :*

$$2(p_{j_0,l_0} - p_{j_1,l_0}) + 2(p_{j_1,l_1} - p_{j_0,l_1}) \neq 0 \; mod \; p \tag{2.21}$$

Ce théorème permet de définir les corollaires suivant la paritś de p.

Corollaire 2.5 *Soit un code QC-LDPC régulier (d_v, d_c) de girth $g \geq 8$. Si p est impair alors il n'existe pas de 8-cycle non-vu formé par 2×2 blocs.*

Preuve : *Pour un 8-cycle formé par 2×2 blocs, nous avons :*

$$2(p_{j_0,l_0} - p_{j_1,l_0}) + 2(p_{j_1,l_1} - p_{j_0,l_1}) = 0 \; mod \; p$$

qui peut s'écrire aussi sous la forme

$$2(p_{j_0,l_0} - p_{j_1,l_0}) + 2(p_{j_1,l_1} - p_{j_0,l_1}) = qp_{q\in\{0,1,2,3\}}$$

Si $q = 0$ ou 2 nous avons :

$$(p_{j_0,l_0} - p_{j_1,l_0}) + (p_{j_1,l_1} - p_{j_0,l_1}) = 0 \; mod \; p$$

qui signifie un 4-cycle. Donc les seuls cas restants sont :

$$2(p_{j_0,l_0} - p_{j_1,l_0}) + 2(p_{j_1,l_1} - p_{j_0,l_1}) = qp_{q\in\{1,3\}}$$

qui n'est possible que si p est pair ; vu que q est impair.

Corollaire 2.6 *Pour p pair, une condition nécessaire et suffisante pour éviter un 8-cycle formé par 2×2 blocs est :*

$$p_{j_1,l_1} \neq \frac{qp - 2(p_{j_0,l_0} - p_{j_1,l_0})}{2} + p_{j_0,l_1} \; mod \; p \;\; avec \; q \in \{1,3\} \tag{2.22}$$

Preuve : *La preuve est identique à la précédente et avec la relation :*

$$2(p_{j_0,l_0} - p_{j_1,l_0}) + 2(p_{j_1,l_1} - p_{j_0,l_1}) = qp_{q\in\{1,3\}}$$

nous en déduisons que :

$$p_{j_1,l_1} \neq \frac{qp-2(p_{j_0,l_0}-p_{j_1,l_0})}{2} + p_{j_0,l_1} \; mod \; p \;\; avec \; q \in \{1,3\}$$

• 8-Cycle non-vu formé par 2×3 blocs

Soit un 8-cycle tel que les blocs colonnes d'indice l_1 soient les seuls blocs traversés deux fois, parmi les blocs qui forment le cycle. En supposant $j_0 = j_2$ et $l_0 \neq l_2$ comme le montre la figure 2.4, alors l'équation 2.19 devient :

$$2(p_{j_1,l_1} - p_{j_0,l_1}) + p_{j_0,l_0} - p_{j_1,l_0} + p_{j_0,l_2} - p_{j_1,l_2} = 0 \; mod \; p \tag{2.23}$$

En se basant sur l'équation 2.23, nous définissons le corollaire suivant.

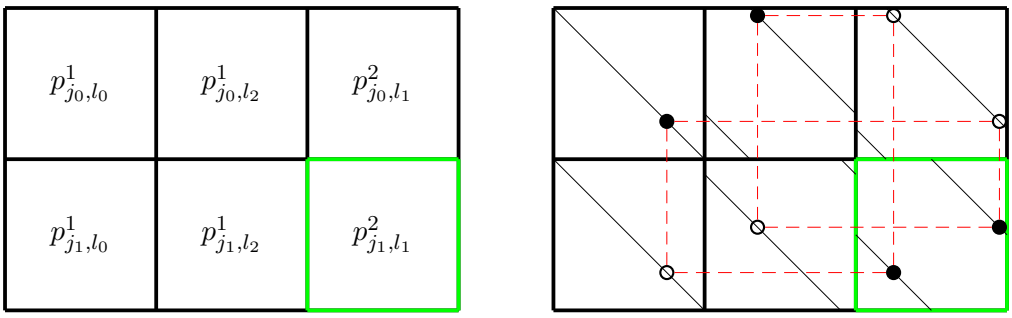


FIGURE 2.4 – 8-cycle formé par 2×3 blocs

Corollaire 2.7 *Une condition nécessaire et suffisante pour éviter un 8-cycle non-vu formé par* 2×3 *blocs est :*

$$p_{j_1,l_1} \neq \frac{qp - p_{j_0,l_0} + p_{j_1,l_0} - p_{j_0,l_2} + p_{j_1,l_2}}{2} + p_{j_0,l_1} \quad avec\ q \in \{0,1,2,3\} \tag{2.24}$$

Preuve : *Le graphe n'admet pas de 8-cycle formé par* 2×3 *blocs si et seulement si :*

$$\begin{aligned} & 2(p_{j_1,l_1} - p_{j_0,l_1}) + p_{j_0,l_0} - p_{j_1,l_0} + p_{j_0,l_2} - p_{j_1,l_2} \neq 0\ mod\ p \\ \Rightarrow \quad & 2(p_{j_1,l_1} - p_{j_0,l_1}) + p_{j_0,l_0} - p_{j_1,l_0} + p_{j_0,l_2} - p_{j_1,l_2} \neq qp\ avec\ q \in \{0,1,2,3\} \\ \Rightarrow \quad & p_{j_1,l_1} \neq \frac{qp - p_{j_0,l_0} + p_{j_1,l_0} - p_{j_0,l_2} + p_{j_1,l_2}}{2} + p_{j_0,l_1}\ avec\ q \in \{0,1,2,3\} \end{aligned}$$

• **8-Cycle non-vu formé par** 3×2 **blocs**

Soit un 8-cycle tel que les blocs lignes d'indice j_1 soient les seuls traversés deux fois parmi les blocs qui forment le cycle. Avec $j_0 \neq j_2$ et $l_0 = l_2$ comme le montre la figure 2.5, alors l'équation 2.19 devient :

$$2(p_{j_1,l_1} - p_{j_1,l_0}) + p_{j_0,l_0} - p_{j_0,l_1} + p_{j_2,l_0} - p_{j_2,l_1} = 0\ mod\ p \tag{2.25}$$

Le 8-cycle représenté sur la figure 2.5 est isomorphe à tous les 8-cycles formés par 3 bloc lignes et 2 bloc colonnes. Un résultat direct de l'équation 2.25 est le corollaire suivant.

Corollaire 2.8 *Une condition nécessaire et suffisante pour éviter un 8-cycle non-vu formé par* 3×2 *blocs est :*

$$p_{j_1,l_1} \neq \frac{qp - p_{j_0,l_0} + p_{j_0,l_1} - p_{j_2,l_0} + p_{j_2,l_1}}{2} + p_{j_1,l_0} \quad avec\ q \in \{0,1,2,3\} \tag{2.26}$$

Preuve : *La preuve est analogue avec celle du corollaire 2.7*

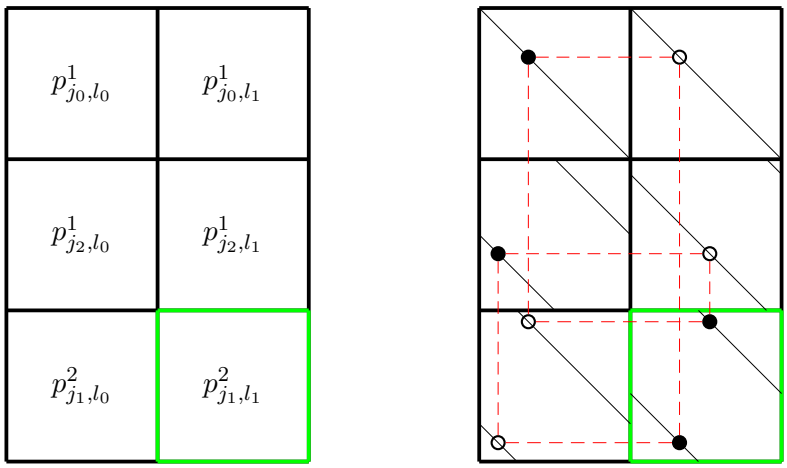


FIGURE 2.5 – 8-cycle formé par 3×2 blocs

• **8-Cycle non-vu formé par 3×3 blocs**

Soit un 8-cycle tel que seul le bloc d'indice j_1, l_1 soit traversé deux fois parmi les blocs qui forment le cycle donc $j_0 \neq j_2$ et $l_0 \neq l_2$ comme le montre la figure 2.6, et l'équation 2.19 devient :

$$2p_{j_1,l_1} + p_{j_0,l_0} - p_{j_0,l_1} - p_{j_1,l_0} - p_{j_1,l_2} + p_{j_2,l_2} - p_{j_2,l_1} = 0 \; mod \; p \tag{2.27}$$

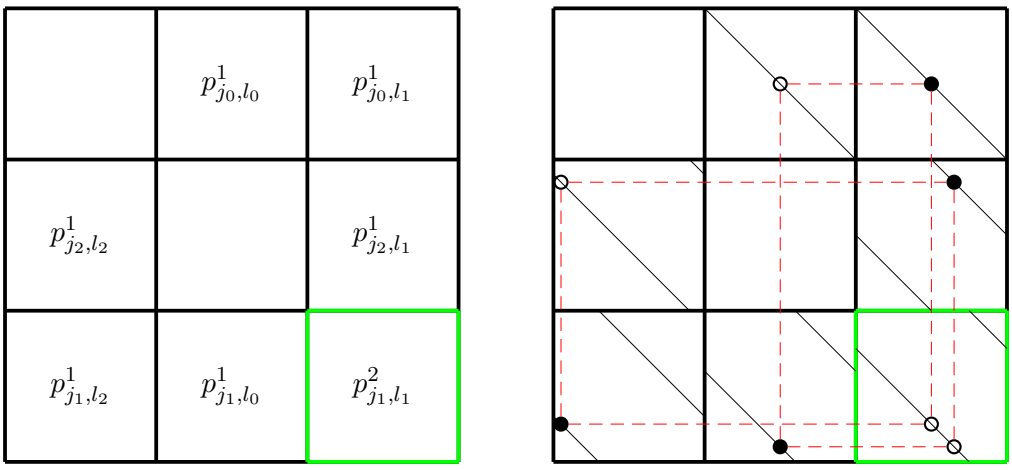


FIGURE 2.6 – 8-cycle formé par 3×3 blocs

Le 8-cycle représenté sur la figure 2.6 est isomorphe à tous les 8-cycles formés par 3 bloc lignes et 3 bloc colonnes où le dernier bloc ligne et colonne est le seul traversé deux fois par le cycle. Ici aussi un résultat direct de l'équation 2.27 est le corollaire suivant.

Corollaire 2.9 *Une condition nécessaire et suffisante pour éviter un 8-cycle non-vu formé par* 3×3 *blocs est :*

$$p_{j_1,l_1} \neq \frac{qp - p_{j_0,l_0} + p_{j_0,l_1} + p_{j_1,l_0} + p_{j_1,l_2} - p_{j_2,l_2} + p_{j_2,l_1}}{2} \quad avec\ q \in \{0,1,2,3\} \quad (2.28)$$

Preuve : *La preuve est aussi analogue avec celle du corollaire 2.7*

Ainsi pour construire un code QC-LDPC de girth $g = 10$ avec l'algorithme RandPEG, il suffit d'appliquer ces conditions définies dans les corollaires 2.6, 2.7, 2.8, et 2.9 lors de la création d'une nouvelle branche. En effet il faut éliminer des potentiels candidats, les nœuds de contrôle correspondants aux valeurs à éviter ; ces valeurs étant calculées à partir des entrées existantes de la matrice de base H_B.

2.4.3 Caractérisation des 10-cycles non-vu

D'après l'équation 2.3, les 10-cycles satisfont à l'équation :

$$\sum_{k=0}^{4} (p_{j_k,l_k} - p_{j_{k+1},l_k}) = 0 \ mod\ p \quad (2.29)$$

avec $j_0 \neq j_1 \neq j_2 \neq j_3 \neq j_4 \neq j_0$ et $l_0 \neq l_1 \neq l_2 \neq l_3 \neq l_4 \neq l_0$. Comme avec les cycles de longueur 8, nous allons expliciter les différents scénarios pouvant conduire à un 10-cycle non-vu pour en déduire par la suite une condition nécessaire et suffisante de les éviter.

Fossorier a montré que les cycles formés par $2 \times q$ blocs ou $q \times 2$ blocs ont des longueurs multiples de 4. De plus, par définition, les cycles non-vus traversent au moins deux fois le nouveau bloc construit. Par conséquent un 10-cycle pouvant être non-vu ne peut pas être formé par $2 \times q$ blocs ou $q \times 2$ ou $5 \times q$ blocs ou $q \times 5$ mais seulement par l'un des cas suivants : 3×3 blocs ou 3×4 blocs ou 4×3 blocs ou 4×4 blocs. Nous conservons la notation $p_{j,l}^{\alpha}$ pour représenter le bloc de la sous matrice $I(p_{j,l})$ dans le cycle, où α est le nombre de fois que le cycle traverse $I(p_{j,l})$. Dans tous les scénarios que nous traitons pour éviter les 10-cycles non-vus, nous supposons que le *nouveau bloc en vert est traversé deux fois et qu'il est celui formé par le dernier bloc ligne et le dernier bloc.* L'indice de ce nouveau bloc est (j_4, l_4) avec $j_4 = j_1$, $l_4 = l_1$

• **Cycle non-vu formé par 3×3 blocs**

Pour les cycles de longueur 10 pouvant être non-vus formés par 3×3 blocs, nous avons trois cas qui ne peuvent pas être déduits par isomorphisme.

◇ **Premier Cas**

Soit un 10-cycle tel que les blocs colonnes d'indice l_1 et les bloc lignes d'indice j_1 soient les seuls traversés deux fois parmi les blocs qui forment le cycle. Donc $j_0 = j_2$ et $l_0 = l_2$ ou $j_0 = j_2$ et $l_0 = l_3$ comme le montre la figure 2.7 où nous supposons $l_0 = l_3$. Dans ce cas, l'équation 2.19 devient :

$$2p_{j_1,l_1} - 2p_{j_0,l_1} + p_{j_0,l_0} + p_{j_0,l_2} - p_{j_3,l_2} + p_{j_3,l_0} - 2p_{j_1,l_0} = 0 \; mod \; p \qquad (2.30)$$

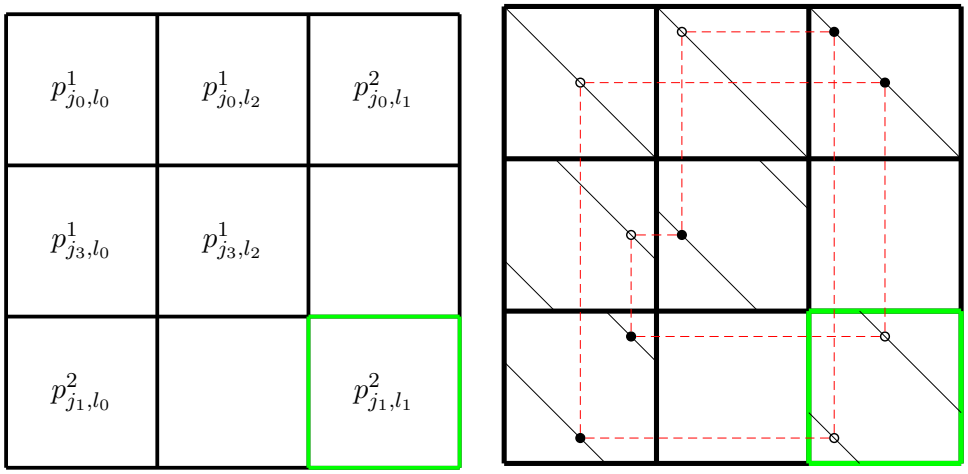


FIGURE 2.7 – Cas 1 : 10-cycle formé par 3×3 blocs

Le 10-cycle représenté sur la figure 2.7 est isomorphe à tous les 10-cycles formés par 3 bloc lignes et 3 bloc colonnes obtenus par permutation de deux premiers bloc lignes ou par permutation des deux premiers bloc colonnes. Un résultat direct de l'équation 2.30 est le corollaire suivant.

Corollaire 2.10 *Une condition nécessaire et suffisante pour éviter un 10-cycle non-vu formé par 3×3 blocs dont la structure est celle de la figure 2.7 est :*

$$p_{j_1,l_1} \neq \frac{qp + 2p_{j_0,l_1} - p_{j_0,l_0} - p_{j_0,l_2} + p_{j_3,l_2} - p_{j_3,l_0} + 2p_{j_1,l_0}}{2}$$
$$avec \; q \in \{0,1,2,3,4\} \qquad (2.31)$$

Preuve : *La preuve est analogue aussi avec celle du corollaire 2.7*

◇ Deuxième Cas

Soit un 10-cycle comme représenté sur la figure 2.8 où nous supposons $l_0 = l_2$ et $j_0 = j_2$. L'équation 2.19 devient :

$$2p_{j_1,l_1} - 2p_{j_0,l_1} + 2p_{j_0,l_0} + p_{j_3,l_3} - p_{j_3,l_0} - p_{j_1,l_0} - p_{j_1,l_3} = 0 \; mod \; p \qquad (2.32)$$

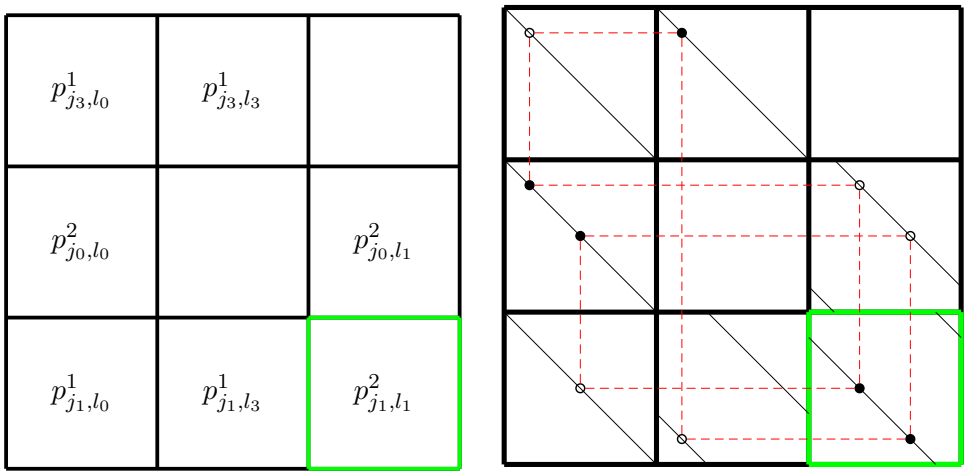


FIGURE 2.8 – Cas 2 : 10-cycle formé par 3×3 blocs

Le 10-cycle représenté sur la figure 2.8 est isomorphe à tous les 10-cycles formés par 3 bloc lignes et 3 bloc colonnes obtenus par permutation de deux premiers bloc lignes ou par permutation des deux premiers bloc colonnes. De même, un résultat direct de l'équation 2.32 est le corollaire suivant.

Corollaire 2.11 *Une condition nécessaire et suffisante pour éviter un 10-cycle non-vu formé par* 3×3 *blocs dont la structure est celle de la figure 2.8 est :*

$$p_{j_1,l_1} \neq \frac{qp + 2p_{j_0,l_1} - 2p_{j_0,l_0} - p_{j_3,l_3} + p_{j_3,l_0} + p_{j_1,l_0} + p_{j_1,l_3}}{2}$$

$$avec \; q \in \{0,1,2,3,4\} \qquad (2.33)$$

◇ Troisième Cas

Soit un 10-cycle comme représenté sur la figure 2.9 où nous supposons $l_0 = l_3$ et $j_0 = j_3$. L'équation 2.19 devient :

$$2p_{j_1,l_1} - 2p_{j_1,l_0} + 2p_{j_0,l_0} - p_{j_0,l_1} - p_{j_0,l_2} + p_{j_2,l_2} - p_{j_2,l_1} = 0 \; mod \; p \qquad (2.34)$$

Le 10-cycle représenté sur la figure 2.9 est isomorphe à tous les 10-cycles formés par 3 bloc lignes et 3 bloc colonnes obtenus par permutation de deux premiers bloc

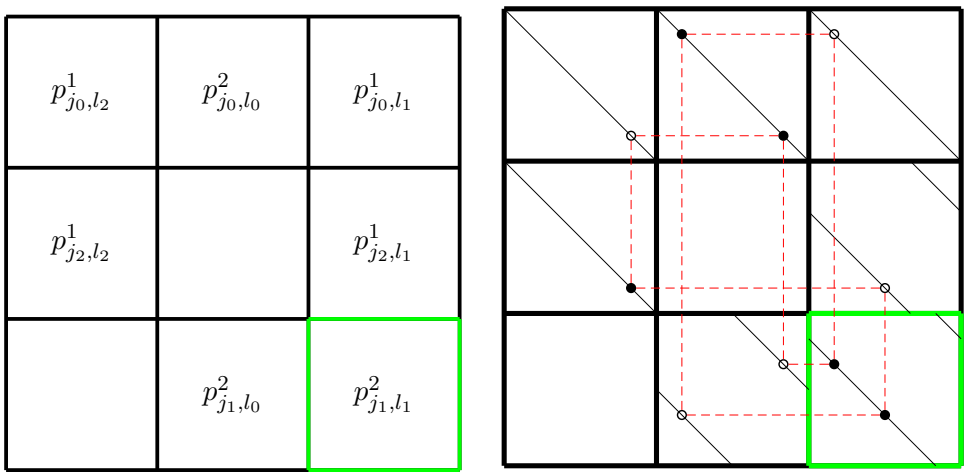


FIGURE 2.9 – Cas 3 : 10-cycle formé par 3×3 blocs

lignes ou par permutation des deux premiers bloc colonnes. De même, un résultat direct de l'équation 2.34 est le corollaire suivant.

Corollaire 2.12 *Une condition nécessaire et suffisante pour éviter un 10-cycle non-vu formé par* 3×3 *blocs dont la structure est celle de la figure 2.9 est :*

$$p_{j_1,l_1} \neq \frac{qp + 2p_{j_1,l_0} - 2p_{j_0,l_0} + p_{j_0,l_1} + p_{j_0,l_2} - p_{j_2,l_2} + p_{j_2,l_1}}{2} \quad avec\ q \in \{0,1,2,3,4\} \quad (2.35)$$

• **Cycle non-vu formé par** 3×4 **blocs**

Pour les 10-cycles pouvant être non-vus formés par 3×4 blocs, nous avons deux cas qui ne peuvent pas être déduits par isomorphisme.

◇ **Premier Cas**

Soit un 10-cycle tel que les blocs colonnes d'indice l_1 soient les seuls traversés deux fois parmi les blocs qui forment le cycle. Donc $j_0 = j_2$ comme le montre la figure 2.10. Dans ce cas l'équation 2.19 devient :

$$2p_{j_1,l_1} - 2p_{j_0,l_1} + p_{j_0,l_0} + p_{j_0,l_2} - p_{j_3,l_2} + p_{j_3,l_3} - p_{j_1,l_0} - p_{j_1,l_3} = 0\ mod\ p \quad (2.36)$$

Le 10-cycle représenté sur la figure 2.10 est isomorphe à tous les 10-cycles formés par 3 bloc lignes et 4 bloc colonnes obtenus par permutation de deux premiers bloc lignes ou par permutation des trois premiers bloc colonnes. Un résultat direct de l'équation 2.36 est le corollaire suivant.

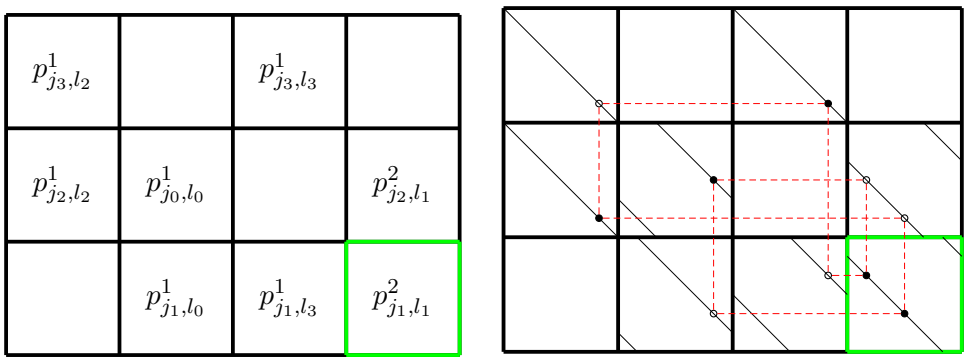


FIGURE 2.10 – Cas 1 : 10-cycle formé par 3×4 blocs

Corollaire 2.13 *Une condition nécessaire et suffisante pour éviter un 10-cycle non-vu formé par* 3×4 *blocs dont la structure est celle de la figure 2.10 est :*

$$p_{j_1,l_1} \neq \frac{qp + 2p_{j_0,l_1} - p_{j_0,l_0} - p_{j_0,l_2} + p_{j_3,l_2} - p_{j_3,l_3} + p_{j_1,l_0} + p_{j_1,l_3}}{2}$$
$$avec\ q \in \{0,1,2,3,4\} \quad (2.37)$$

◇ Deuxième Cas

Soit un 10-cycle tel que seul le bloc d'indice j_1, l_1 soit traversé deux fois parmi les blocs qui forment le cycle comme le montre la figure 2.11. En supposant $j_0 = j_3$, l'équation 2.19 devient :

$$2p_{j_1,l_1} - p_{j_0,l_1} + p_{j_0,l_0} + p_{j_0,l_3} - p_{j_0,l_2} + p_{j_2,l_2} - p_{j_2,l_1} - p_{j_1,l_3} - p_{j_1,l_0} = 0\ mod\ p \quad (2.38)$$

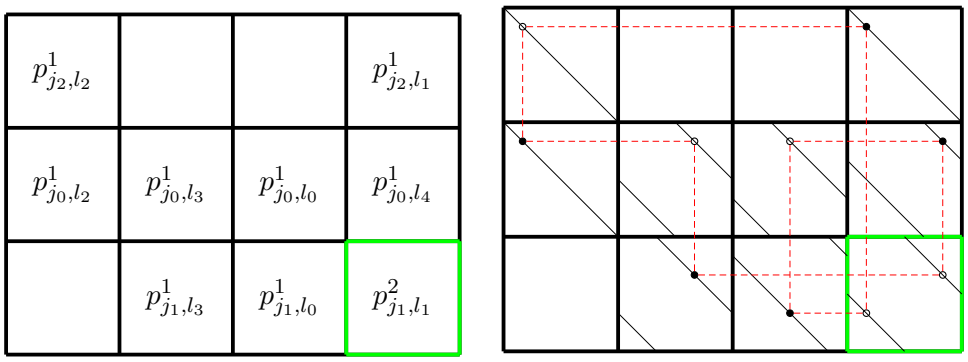


FIGURE 2.11 – Cas 2 : 10-cycle formé par 3×4 blocs

Le 10-cycle représenté sur la figure 2.11 est isomorphe à tous les 10-cycles formés par 3 bloc lignes et 4 bloc colonnes obtenus par permutation de deux premiers bloc lignes ou par permutation des trois premiers bloc colonnes. Un

résultat direct de l'équation 2.38 est le corollaire suivant.

Corollaire 2.14 *Une condition nécessaire et suffisante pour éviter un 10-cycle non-vu formé par* 3×4 *blocs dont la structure est celle de la figure 2.11 est :*

$$p_{j_1,l_1} \neq \frac{qp + p_{j_0,l_1} - p_{j_0,l_0} - p_{j_0,l_3} + p_{j_0,l_2} - p_{j_2,l_2} + p_{j_2,l_1} + p_{j_1,l_3} + p_{j_1,l_0}}{2}$$
$$avec q \in \{0,1,2,3,4\} \quad (2.39)$$

• **Cycle non-vu formé par** 4×3 **blocs**

Pour les 10-cycles pouvant être non-vus formés par 4×3 blocs, nous avons deux cas qui ne peuvent pas être déduits par isomorphisme.

◇ **Premier Cas**

Soit un 10-cycle tel que seul le bloc d'indice j_1, l_1 soit traversé deux fois parmi les blocs qui forment le cycle comme le montre la figure 2.12 alors l'équation 2.19 devient :

$$2p_{j_1,l_1} - p_{j_0,l_1} + p_{j_0,l_0} + p_{j_2,l_0} - p_{j_3,l_0} + p_{j_3,l_3} - p_{j_1,l_0} - p_{j_1,l_3} - p_{j_2,l_1} = 0 \; mod \; p \quad (2.40)$$

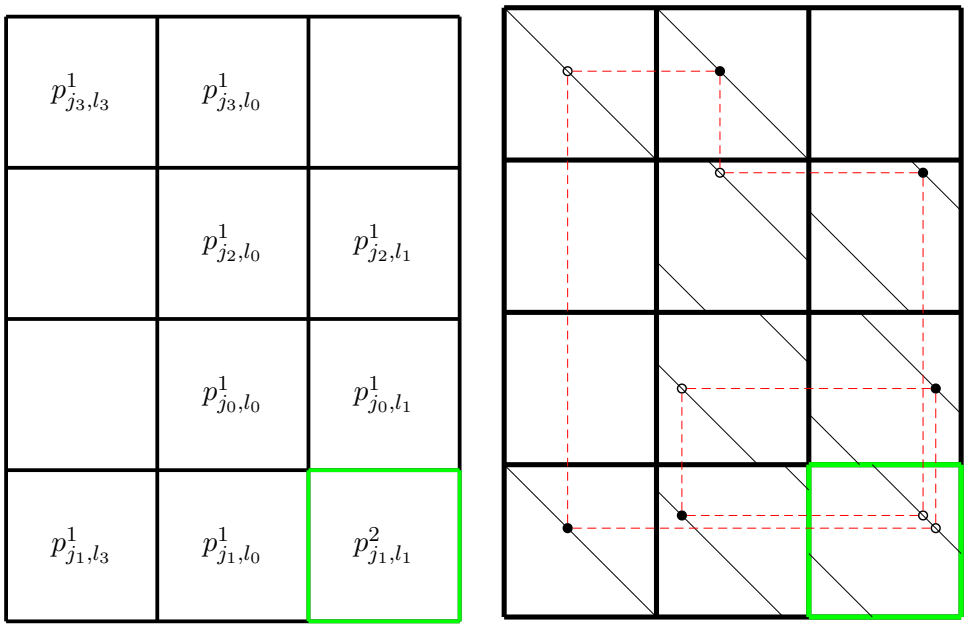


FIGURE 2.12 – Cas 1 : 10-cycle formé par 4×3 blocs

Le 10-cycle représenté sur la figure 2.12 est isomorphe à tous les 10-cycles formés par 4 bloc lignes et 3 bloc colonnes obtenus par permutation de trois

premiers bloc lignes ou par permutation des deux premiers bloc colonnes. Un résultat direct de l'équation 2.40 est le corollaire suivant.

Corollaire 2.15 *Une condition nécessaire et suffisante pour éviter un 10-cycle non-vu formé par* 4×3 *blocs dont la structure est celle de la figure 2.12 est :*

$$p_{j_1,l_1} \neq \frac{qp + p_{j_0,l_1} - p_{j_0,l_0} - p_{j_2,l_0} + p_{j_3,l_0} - p_{j_3,l_3} + p_{j_1,l_0} + p_{j_1,l_3} + p_{j_2,l_1}}{2}$$
$$avec\ q \in \{0,1,2,3,4\} \quad (2.41)$$

◇ Deuxième Cas

Soit un 10-cycle tel que les bloc lignes d'indice j_1 soient les seuls traversés deux fois parmi les blocs qui forment le cycle comme le montre la figure 2.13. En supposant $l_0 = l_3$ sans perdre la généralité, l'équation 2.19 devient :

$$2p_{j_1,l_1} - p_{j_0,l_1} + p_{j_0,l_0} + p_{j_2,l_2} - p_{j_2,l_1} + p_{j_3,l_0} - p_{j_3,l_2} - 2p_{j_1,l_0} = 0\ mod\ p \quad (2.42)$$

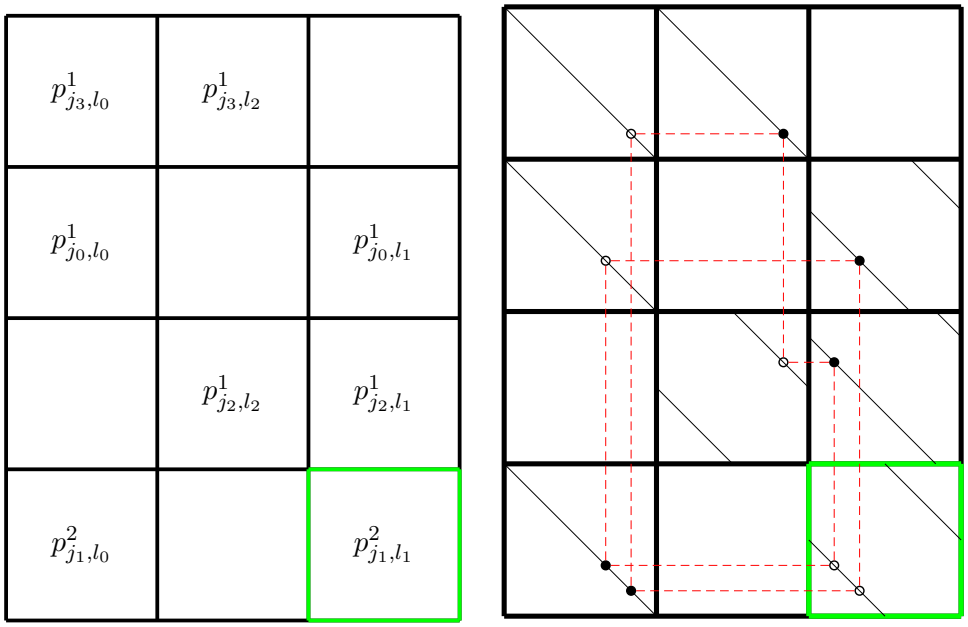


FIGURE 2.13 – Cas 2 : 10-cycle formé par 4×3 blocs

Le 10-cycle représenté sur la figure 2.13 est isomorphe à tous les 10-cycles formés par 4 bloc lignes et 3 bloc colonnes obtenus par permutation de trois premiers bloc lignes ou par permutation des deux premiers bloc colonnes. Un résultat direct de l'équation 2.42 est le corollaire suivant.

Corollaire 2.16 *Une condition nécessaire et suffisante pour éviter un 10-cycle non-vu formé par* 4×3 *blocs dont la structure est celle de la figure 2.13 est :*

$$p_{j_1,l_1} \neq \frac{qp + p_{j_0,l_1} - p_{j_0,l_0} + p_{j_2,l_1} - p_{j_2,l_2} + p_{j_3,l_2} - p_{j_3,l_0} + 2p_{j_1,l_0}}{2}$$
$$avec\ q \in \{0,1,2,3,4\} \quad (2.43)$$

• **Cycle non-vu formé par** 4×4 **blocs**

Pour les 10-cycles pouvant être non-vus formés par 4×4 blocs, nous avons un seul cas possible. Pour ce cas, nous supposons $j_1 = j_3 \neq j_4$ et $l_1 = l_3 \neq l_4$.
Soit un 10-cycle tel que le bloc d'indice j_1, l_1 soit le seul traversé deux fois parmi les blocs qui forment le cycle comme le montre la figure 2.14, l'équation 2.19 devient :

$$2p_{j_1,l_1} - p_{j_0,l_1} + p_{j_0,l_0} + p_{j_2,l_2} - p_{j_1,l_2} + p_{j_4,l_4} - p_{j_0,l_4} - p_{j_4,l_1} - p_{j_2,l_1} = 0\ mod\ p \quad (2.44)$$

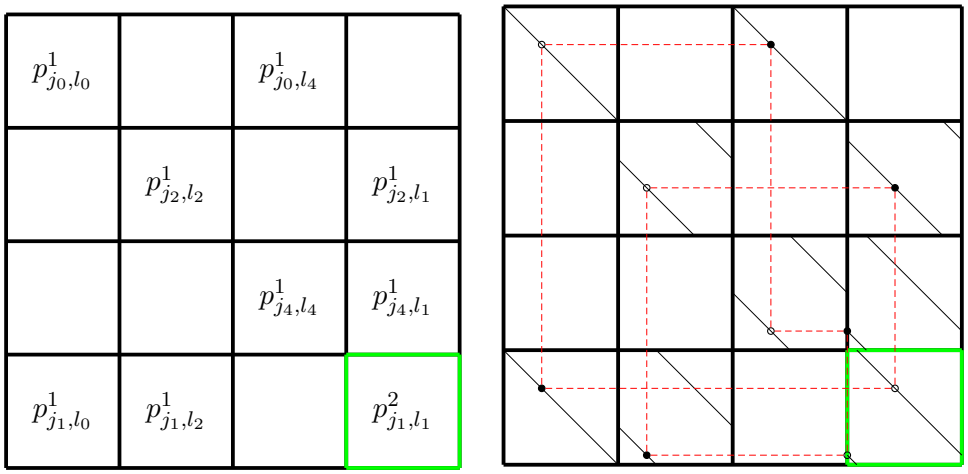


FIGURE 2.14 – 10-cycle formé par 4×4 blocs

Le 10-cycle représenté sur la figure 2.14 est isomorphe à tous les 10-cycles formés par 4 bloc lignes et 4 bloc colonnes obtenus par permutation de trois premiers bloc lignes et/ou par permutation des trois premiers bloc colonnes. Un résultat direct de l'équation 2.44 est le corollaire suivant.

Corollaire 2.17 *Une condition nécessaire et suffisante pour éviter un 10-cycle non-vu formé par* 4×4 *blocs dont la structure est celle de la figure 2.14 est :*

$$p_{j_1,l_1} \neq \frac{qp + p_{j_0,l_1} - p_{j_0,l_0} + p_{j_1,l_2} - p_{j_2,l_2} + p_{j_0,l_4} - p_{j_4,l_4} + p_{j_4,l_1} + p_{j_2,l_1}}{2}$$
$$avec\ q \in \{0,1,2,3,4\} \quad (2.45)$$

Ayant toutes les conditions nécessaires et suffisantes pour éviter les cycles non-vus de longueurs 8 ou 10, nous pouvons les appliquer sur l'algorithme RandPEG pour construire de manière prédictive des codes QC-LDPC de girth $g = 10$ ou $g = 12$.

2.5 Algorithme RandPEG amélioré pour la conception de codes QC-LDPC

L'ensemble des inégalités définies dans les corollaires 2.6 à 2.17 représentent les contraintes sur le facteur de décalage p_{j_1,l_1} pour éviter les 8-cycles et 10-cycles non-vus avec une nouvelle matrice de permutation circulante. Ces contraintes peuvent être réparties en deux sous-ensembles comme décrit sur le tableau 2.1, où $\mathcal{C}o_8$ regroupe les contraintes pour les 8-cycles non-vus et $\mathcal{C}o_{10}$ regroupe les contraintes pour des 10-cycles non-vus.

sous-groupes	contraintes
$\mathcal{C}o_8$	$= \left\{ \begin{array}{l} equation\ 2.22, \\ equation\ 2.24, \\ equation\ 2.26, \\ equation\ 2.28 \end{array} \right\}$
$\mathcal{C}o_{10}$	$= \left\{ \begin{array}{l} equation\ 2.31, \\ equation\ 2.33, \\ equation\ 2.35, \\ equation\ 2.37, \\ equation\ 2.39, \\ equation\ 2.41, \\ equation\ 2.43, \\ equation\ 2.45 \end{array} \right\}$

TABLE 2.1 – Les valeurs donnant des cycles non-vus

Ces contraintes sont intégrées dans une nouvelle fonction coût supplémentaire qui calcul toutes les valeurs interdites pour le facteur de décalage p_{j_1,l_1}. Pour chaque nœud de variable $v_{l.p}$, après l'expansion de l'arbre des voisins, la fonction de coût supplémentaire est exécutée dans le cas où $\overline{\mathcal{M}}^k_{v_n} \neq \varnothing$ et dans le cas où $\overline{\mathcal{M}}^k_{v_n} = \varnothing$ à la différence de la fonction coût du RandPEG original qui n'est exécuté que dans le cas où $\overline{\mathcal{M}}^k_{v_n} = \varnothing$. L'exécution de la fonction coût supplémentaire dans

le cas où $\overline{\mathcal{M}}_{v_n}^{k} \neq \varnothing$ permet de maximiser le girth local et de ne pas créer de cycle de longueur g_t. Cette exécution assure qu'il n'existe pas des 8-cycles et des 10-cycles si le girth ciblé est $g = 10$ ou $g = 12$. L'exécution de la fonction coût supplémentaire dans le cas où $\overline{\mathcal{M}}_{v_n}^{k} = \varnothing$ permet d'éviter des cycles formés par peu de blocs par exemple 2×2 blocs si g=8, ceux qui est avantageux pour un décodeur itératif. Cette fonction coût a un avantage considérable sur la complexité car nous n'avons plus besoin de procéder à p vérifications après le choix d'un nouveau facteur de décalage c'est à dire la selection d'un nouveau nœud de contrôle pour le nœud de variable $v_{l.p}$.

L'algorithme RandPEG avec la fonction coût supplémentaire constitue l'une des principales contributions dans cette thèse. Cet algorithme RandPEG amélioré est décrit à la page suivante.

Remarque 2.2 *Les conditions définies sur les corollaires 2.6 à 2.17 de même que l'algorithme RandPEG modifié-3 restent valables sur les* **protographes binaires type-I**. *Si la protomatrice est définie, alors le RandPEG permet la détermination des décalages. Chaque décalage correspond à l'indice m modulo p du nœud de contrôle c_m choisi parmi les potentiels candidats. Il faut noter que les nœuds de contrôle associés aux entrées de valeurs "0" de la protomatrice sont supprimés des potentiels candidats avant la sélection. Si la protomatrice n'est pas définie, alors le RandPEG permet la détermination des décalages qui définissent par la suite la position des "1" de la protomatrice.*

Input : d_c, p, D_v, g_t
Output : G
for $n = 0$ **to** $d_c(p-1)$ **by gap** p **do**
  for $i = 0$ **to** $d_{v_n} - 1$ **do**
    if *i=0 or n=0* **then**
      $E^0_{v_n} \leftarrow branche\,(c_{i.p}, v_n)$, met la matrice identité sur la première ligne et sur la première colonne
    end
    else
      Déplier, avec les branches existantes, le sous graphe depuis le nœud de variable v_n jusqu'à la profondeur $k = g_t/2 - 1$
      if $\overline{\mathcal{M}}^k_{v_n} \neq \varnothing$ **then**
        if $g_t \geq 8$ **then**
          Retire de $\overline{\mathcal{M}}^k_{v_n}$ tous les candidats qui ne satisfont pas les contraintes $\mathcal{C}o_8$
          if $g_t \geq 12$ **then**
            Retire de $\overline{\mathcal{M}}^k_{v_n}$ tous les candidats qui ne satisfont pas les contraintes $\mathcal{C}o_{10}$
          end
        end
        Mettre à jour $\overline{\mathcal{M}}^k_{v_n}$
      end
      if $\overline{\mathcal{M}}^k_{v_n} \neq \varnothing$ **then**
        Choix aléatoire d'un nœud de contrôle c_m dans $\overline{\mathcal{M}}^k_{v_n}$:
        $E^i_{v_n} \leftarrow branche\,(c_m, v_n)$ (pas de cycle de longueur g_t créé)
      end
      if $\overline{\mathcal{M}}^k_{v_n} = \varnothing$ **then**
        if $\mathcal{M}^k_{v_n} \neq \varnothing$ **then**
          Etape 1 : Elimine les nœuds de profondeur inférieure à k
          Etape 2 : Elimine tous les nœuds de contrôle qui créent plus de cycles que la valeur minimale des cycles $min\,(nbCycles_{c_m})$
          Etape 3 : Elimine tous les nœuds de contrôle qui donnent des cycles non-vus suivant le girth
          if $g_t \geq 10$ **then**
            Retire de $\overline{\mathcal{M}}^k_{v_n}$ tous les candidats qui ne satisfont pas les contraintes $\mathcal{C}o_8$
            if $g_t \geq 12$ **then**
              Retire de $\overline{\mathcal{M}}^k_{v_n}$ tous les candidats qui ne satisfont pas les contraintes $\mathcal{C}o_{10}$
            end
          end
          Etape 4 : Mettre à jour $\mathcal{M}^k_{v_n}$
        end
        if $\mathcal{M}^k_{v_n} \neq \varnothing$ **then**
          Choix aléatoire d'un nœud de contrôle c_m dans $\mathcal{M}^k_{v_n}$:
          $E^i_{v_n} \leftarrow branche\,(c_m, v_n)$
        end
        else
          Echec de l'algorithme
        end
      end
    end
  end
end

Algorithme 3: Algorithme RandPEG QC-LDPC Modifié

Pour évaluer l'efficacité de notre nouvel algorithme, nous procédons à une étude comparative avec les meilleurs algorithmes et méthodes de constructions existantes.

2.6 Etude Comparative de l'algorithme RandPEG avec d'autres méthodes

Nous présentons maintenant une comparaison du RandPEG avec les méthodes de constructions existantes. Pour les codes QC-LDPC où $d_v = 3$, la comparaison se fait avec les algorithmes de recherche aléatoires : le *Hill-Climbing* [WYD08, WDY13] et le *Guest-and-Test* [Fos04]. Pour les codes QC-LDPC où $d_v > 3$, la comparaison est faite avec les méthodes explicites proposées par Zhang et al. [ZSW13b, ZSW13a, ZSW12a].

2.6.1 Comparaison des codes QC-LDPC de girth $g = 8, 10, 12$ avec $d_v = 3$

Parmi les méthodes de constructions basées sur un algorithme de recherche aléatoire nous avons l'algorithme *Guest-and-Test* développé par Fossorier dans [Fos04]. Le principe de fonctionnement du *Guest-and-Test* pour construire un code QC-LDPC régulier (d_v, d_c) de girth g est de rechercher une matrice de base H_B de dimension $d_v \times d_c$ dont les valeurs vérifient l'équation 2.3. Pour déterminer la matrice avec la plus petite valeurs de p, l'algorithme procède par des tests en commençant par $p = (d_v - 1)(d_c - 1) + 1$. Pour chaque test, les entrées de la première ligne et de la première colonne de H_B sont fixées à zéro et les autres entrées sont choisies indépendamment et uniformément entre 0 et $p-1$ de manière à respecter l'équation 2.3. Noter que l'echec pour un p n'implique pas qu'il n'existe pas de code QC-LDPC régulier (d_v, d_c) de longueur $N = p.d_c$.

Wang et al [WYD08, WDY13] ont introduit un autre algorithme de recherche aléatoire, le "*Hill-Climbing*" qui présente actuellement les meilleures valeurs minimales de p pour les codes QC-LDPC. La construction de la matrice de base peut se résumer en quatre étapes :

1. 1^{ere} **Etape :** Génération aléatoire d'une matrice de base $d_v \times d_c$ avec des entrées différentes deux à deux comprises entre "0" et p.
2. 2^{eme} **Etape :** Calcul de la matrice intermédaire de coût C, dont chaque coefficient $c_{j,l}$ minimise le nombre de contraintes non satisfaites dans H_B si on attribue à $p_{j,l}$ chaque valeur entière de 0 à $p - 1$. Ceci revient à chercher

et à choisir parmi toutes les p valeurs comprises entre 0 et $p-1$ celle qui satisfait au plus l'equation 2.3 avec H_B.

3. 3^{eme} **Etape :** Mise à jour de la matrice H_B et reprise de l'étape 2, jusqu'à l'arrêt d'évolution des valeurs de la matrice C.
4. 4^{eme} **Etape :** Si les toutes les valeurs de C sont nulles ; c'est à dire toutes les valeurs de H_B satisfont à l'équation 2.3, alors l'algorithme renvoie la matrice H_B. Sinon il y'a échec de construction pour le girth souhaité avec les paramètres d_v et d_c.

Dans le tableau 2.2 et 2.3, nous comparons les valeurs minimales de p obtenues avec les différents algorithmes. Pour les codes de girth 10, Fossorier n'a pas donné de valeurs minimales. Nous fournissons aussi les valeurs minimales obtenues pour les codes QC-LDPC de girth $g = 12$, dont les seuls résultats vus dans la littérature sont ceux de O'Sullivan [O'S06]. Il propose une méthode de construction algébrique dont ses résultats minimaux sont similaires à ceux du Guest-and-Test pour les codes de girth 8 mais pour les codes de girth $g = 10$ les résultats dans [O'S06] sont très grands par rapport aux résultats du Hill-Climbing.

d_c	4	5	6	7	8	9	10	11	12
Guest-and-Test (Table II) [Fos04]	9	14	18	21	26	33	39	46	54
Hill-Climbing (Table I) [WDY13]	9	13	18	21	25	30	35	41	47
RandPEG	9	13	18	21	25	30	35	40	46
Borne conjecture	9	13	17	21	25	29	33	37	41
Meilleure borne minimale eq. 2.16	9	12	15	18	21	24	27	30	33

TABLE 2.2 – Comparaison valeur Minimale de p pour $d_v = 3$, $g = 8$; en rouge les résultats identiques aux bornes et en bleu les meilleurs résultats différents des bornes

d_c	4	5	6	7	8	9	10	11
Hill-Climbing (Table II) [WDY13]	39	63	103	160	233	329	439	577
RandPEG	37	61	91	155	227	323	429	571
Borne Karimi et al. [KB13], eq. 2.12	37	61	91	127	168	217	271	331

TABLE 2.3 – Comparaison valeur Minimale de p pour $d_v = 3$, $g = 10$; en rouge les résultats identiques aux bornes et en bleu les meilleurs résultats différents des bornes

d_c	4	5	6	7	8	9
O'Sullivan (Table V) [O'S06]	97	239	479	881	1493	2087
RandPEG	73	163	369	679	1291	1963
Borne minimale equation 2.13	43	73	111	157	211	273

TABLE 2.4 – Comparaison valeur Minimale de p pour $d_v = 3$, $g = 12$; en rouge les résultats identiques aux bornes et en bleu les meilleurs résultats différents des bornes

Pour les codes de girth $g = 8$ nous remarquons sur le tableau 2.2 que le RandPEG et le Hill-Climbing ont quasi les mêmes valeurs minimales et ces valeurs en rouge sont les valeurs optimales définies à la conjecture 2.1 pour $d_c \leq 8$. Pour les codes de girth $g = 10$ et $g = 12$ les cycles non-vus interviennent et les résultats des valeurs minimales sont donnés sur les tableaux 2.3 et 2.4. *Pour les codes QC-LDPC de girth 10 seul notre algorithme RandPEG atteint les bornes minimales définies par Karimi et Banihashemi [KB13] et avec des taux de succès importants pour* $4 \leq d_c \leq 6$. *Ces résultats montrent l'efficacité de notre algorithme.* Pour les codes de girth $g = 12$, les valeurs minimales obtenues avec le RandPEG sont meilleures que les résultats obtenus par O'Sullivan. En effet, pour chaque d_c la valeur avec le RandPEG presque égale au 3/4 de la valeur donnée par O'Sullivan. Il faut noter que de manière générale si d_c augmente, alors la valeur minimale de p croît considérablement et parallélement les taux de succès décroissent. Ces deux résultats font que les valeurs de p qui garantissent des codes QC-LDPC large girth peuvent être très élevées si d_c est grand.

2.6.2 Comparaison des codes QC-LDPC de girth $g = 6, 8$ avec $d_v > 3$

Pour $d_v > 3$ et $g = 8$, les bornes minimales proposées dans la littérature sont généralement faibles. Les meilleurs résultats sont proposés par des solutions constructives explicites. Pour ces méthodes de constructions explicites les bornes minimales des codes QC-LDPC de girth $g = 6$ sont données par l'équation suivante qui est une conséquence du théorème 2.4,

$$p \geq d_c + ((d_c + 1) \, mod \, 2) \tag{2.46}$$

Pour les codes de girth $g = 8$ les valeurs minimales de p sont données par les relations suivantes :

– $p \geq d_c\left(d_c + [d_c\,(mod\,2)]\right)/2 + 1$, pour les codes réguliers $(3, d_c)$ [ZSW13b].
– $p \geq 3d_c^2/4 + d_c - 1$, pour les codes réguliers $(4, d_c)$ [ZSW12a].
– $p \geq (2d_c + 3)(d_c - 1) + 1$, pour les codes réguliers $(5, d_c)$ [ZZ14].
– $p \geq 2(d_c + 5)(d_c - 1) + 1$, pour les codes réguliers $(6, d_c)$ [ZZ14].

Ces valeurs minimales ainsi que les résultats minimaux obtenus avec le RandPEG sont représentés dans les tableaux 2.5 et 2.6.

d_c	5	6	7	8	9	10	11	12
Borne minimale avec $d_v = 4, 5$	5	7	7	9	9	11	11	13
RandPEG avec $d_v = 4$	5	7	7	10	10	11	11	13
RandPEG avec $d_v = 5$	-	7	7	10	10	11	11	13

TABLE 2.5 – Bornes et valeurs minimales de p pour $g = 6$, en rouge les résultats identiques aux bornes

Méthode \ d_c	5	6	7	8	9	10	11	12
explicite avec $d_v = 4$	23	32	43	55	69	84	101	119
RandPEG avec $d_v = 4$	20	24	37	49	59	75	87	105
explicite avec $d_v = 5$	-	76	103	134	169	208	251	298
RandPEG avec $d_v = 5$	-	47	63	84	103	127	157	179

TABLE 2.6 – Comparaison des valeurs minimales de p pour $g = 8$, en blue les meilleurs résultats

Pour les codes de girth $g = 6$, à l'exception de $d_c = 8$ et $d_c = 9$, le RandPEG atteint les bornes minimales définies par l'équation 2.46. La particularité de ces résultats est que p varie uniquement en fonction de d_c.

Pour les codes de girth $g = 8$, les résultats minimaux obtenus avec le RandPEG sont meilleurs que ceux des méthodes explicites. Le tableau 2.6 montre que pour d_v fixé, plus d_c augmente plus la différence entre les résultats du RandPEG et ceux des constructions explicites est importante. Idem pour d_c fixé, plus d_v augmente plus la différence entre les résultats minimaux est importante. Par exemple pour chaque d_c, la valeur minimale obtenue avec le RandPEG est le 4/5 de la valeur minimale obtenue avec la construction explicite pour $d_v = 4$ tandis que pour

$d_v = 5$ la valeur du RandPEG est le 3/5 de la valeur de construction explicite.

Pour mieux prouver l'efficacité du RandPEG par rapport aux méthodes explicites, en plus des valeurs minimales plus faibles, nous évaluons le taux de succès pour $d_c = 9$ et $d_c = 12$ pour lesquels les valeurs minimales de p sont 46 et 73 avec les méthodes explicites. Nous voyons sur les figures 2.15 et 2.16 un taux de succès supérieur à 50% à partir de $p = 43$ pour $d_c = 9$ (figure 2.15) et au delà de $p = 69$ pour $d_c = 12$ (figure 2.16). Ces résultats montrent que des codes QC-LDPC avec de bon rendement peuvent être construits avec le RandPEG.

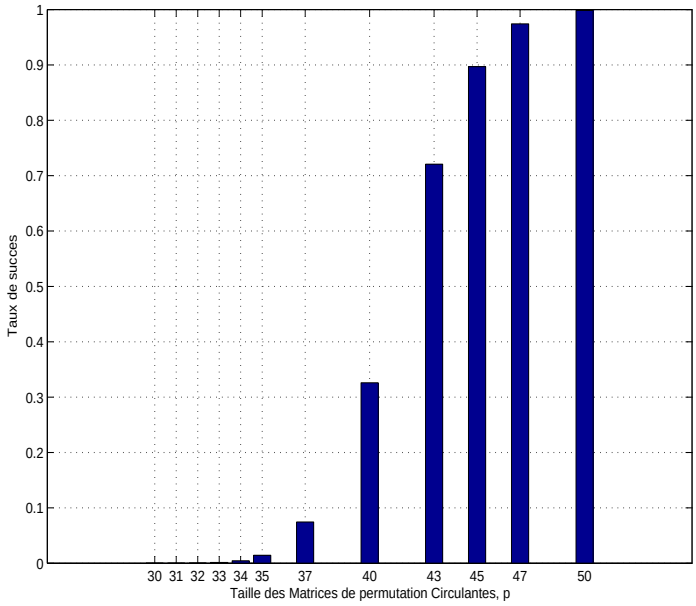


FIGURE 2.15 – Taux de succès en fonction de p pour $d_v = 3$, $d_c = 9$ et $g = 8$

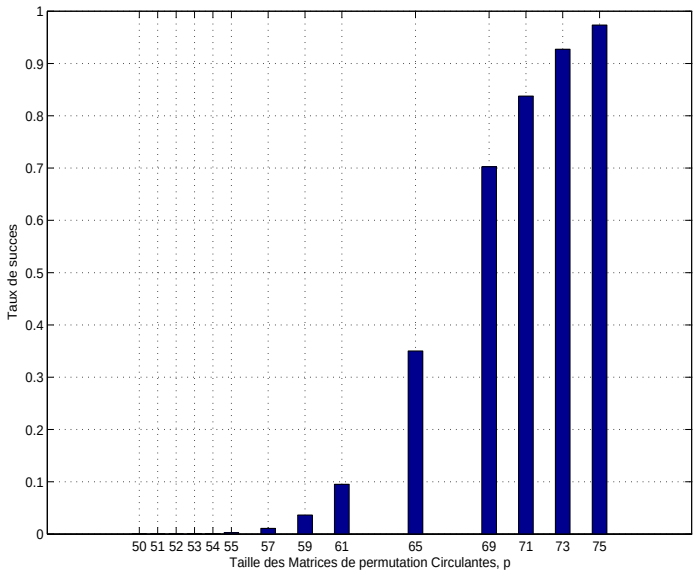


FIGURE 2.16 – Taux de succès en fonction de p pour $d_v = 3$, $d_c = 12$ et $g = 8$

2.7 Conclusion

Dans ce chapitre nous avons présenté les codes QC-LDPC avec des critères sur les entrées de la matrice de base pour obtenir un girth voulu. A partir de ces critères nous avons proposé une conjecture sur la valeur minimale p_{min} de la taille des matrices de permutation circulantes pour les codes QC-LDPC réguliers $(3, d_c)$ de girth 8. Par la suite nous avons défini des conditions nécessaires et suffisantes pour éviter les cycles non-vus avec les algorithmes du type PEG pour QC-LDPC. Ces conditions sont appliquées sur l'algorithme RandPEG ce qui permet d'éviter des vérifications à postériori et ainsi réduire la complexité durant la construction d'un code. Pour valider l'efficacité de ce nouveau algorithme des comparaisons avec les meilleurs algorithmes et les meilleures méthodes de conceptions ont été effectuées. Notre algorithme RandPEG atteint les bornes minimales définies pour les codes de girth $g = 6$. Pour les codes de girth $g = 8$, avec $d_v = 3$ nous atteignons les bornes définies dans notre conjecture pour $d_c \leq 8$ et avec $d_v > 3$ notre algorithme donne de meilleurs résultats que ceux proposés par les méthodes de constructions explicites. Pour les codes de girth $g = 10$ seul notre algorithme parmi ceux étudiés permet d'atteindre les bornes optimales définies. De manière générale notre algorithme, qui est une des principales contributions dans cette thèse, donne les meilleurs résultats minimaux de p, permettant ainsi une construction optimale de code QC-LDPC.

Cependant, pour les codes QC-LDPC de girth $g \geq 8$, il faut noter que la borne minimale p croît considérablement en fonction de d_c avec un taux de succès de l'algorithme faible pour les bornes minimales de d_c important. Ainsi pour des codes de girth $g \geq 10$, cette évolution entraine des codes de longueurs importantes pour des paramètres d_v et d_c élevés. Or les codes de longueurs importantes ne sont pas adaptés aux applications pratiques donc il faut trouver des moyens d'éviter les combinaisons de cycles courts. C'est l'objectif du chapitre suivant.

Chapitre 3

Conception de codes LDPC sans trapping-sets

Dans ce chapitre, nous présentons notre deuxième contribution principale qui est un algorithme de construction de code LDPC sans trapping-set/expansionsets. Nous allons utiliser à nouveau le RandPEG et le modifier pour prendre en compte une caractérisation des trapping-sets.

La conception de codes LDPC binaires avec un faible plancher d'erreurs est encore un problème considérable non entièrement résolu dans la littérature. Pour les canaux bruités tel que le canal à bruit blanc gaussien additif (AWGN : Additive White Gaussian Noise) ou le canal binaire symétrique (CBS ou BSC : Binary Symmetric Channel), le plancher d'erreurs des décodeurs LDPC itératifs est dû à la présence des structures appelées *"trapping-sets"* [Ric03]. Pour les canaux binaires à effacement (CBE ou BEC : Binary Erasure Channel), ces trapping-sets sont connus sous la notion de *"stopping-sets"* [DPT$^+$02].

Les trapping-sets peuvant être considérés comme une combinaison de plusieurs cycles de longueur faible, une solution pour les éviter est de construire des codes LDPC de large girth. Dans le cas des applications pratiques où la longueur et le rendement sont fixées, le girth ne peut pas être augmenté. La solution est alors d'identifier les trapping-sets les plus nocifs et de les éviter [DZA$^+$07, OUVZ02, CNVM10b]. C'est cette solution que nous développons dans ce chapitre en proposant un algorithme dont sa particularité est que la détection des topologies, trapping-sets ou expansion-sets, se fait de manière prédictive à partir de l'arbre des voisins en tenant compte des cycles non-vus pour les codes QCLDPC.

Dans un premier temps nous faisons une brève description des structures particulières avec leurs notations sous forme de préliminaires. Dans un second temps,

nous présentons la caractérisation et les critères de détection de quelques trapping-sets. Cette présentation intégre une description de nouveau algorithme dont son efficacité est démontrée par des comparaisons en termes de performances de correction d'erreurs avec des codes existants. Dans un dernier temps nous généralisons les critères de détections avec les expansions-sets.

3.1 Préliminaires et Notations

Soit G le graphe de Tanner d'un code LDPC binaire $\mathcal{C}$(N,K) de rendement $R = K/N$. Ce graphe est constitué d'un ensemble de N nœuds de variable $V = \{v_0, v_1, \cdots, v_{N-1}\}$ et d'un ensemble de M nœuds de contrôle $C = \{c_0, c_1, \cdots, c_{M-1}\}$. Un *l-cycle* ou cycle de longueur l dans G une suite alternative de nœuds de variable et de nœuds de contrôle $v_{n_0}, c_{m_0}, v_{n_1}, c_{m_1}, \ldots, v_{n_{l/2}}$, avec $v_{n_0} = v_{n_{l/2}}$. Ces cycles sont parmi les premiers facteurs d'échec des décodeurs itératifs. Richarson a fait une étude sur les bits erronés que le décodeur ne corrigeait pas à l'issu des itérations. Il introduit alors la notion de *trapping-sets* dans [Ric03], qu' il définit comme suit :

Définition 3.1 : *Pour un décodeur itératif donné d'entrée* $\mathbf{y}$*, un* **trapping-set** *(TS) dénoté par* $\mathbf{T}(\mathbf{y})$ *est un ensemble non-vide de nœuds de variable dans G qui n'est pas corrigé après un nombre infini d'itérations. Le décodeur est "trapped".*

Suivant le canal de transmission, l'algorithme de décodage du décodeur et la configuration des voisins des nœuds de variable du trapping-set, les trapping-sets sont appelés : stopping-sets [DPT+02], absorbing-sets [DZA+10], ou trapping-sets [Ric03].

3.1.1 Stopping-Sets (SS)

Die et al [DPT+02] définissent un stopping-set comme un ensemble de "s" nœuds de variable noté V_s, où tous les voisins de ces "s" nœuds de variable sont au moins connectés deux fois sur l'ensembe V_s. Une autre définition plus adaptée au décodeur à effacement est la suivante :

Définition 3.2 : *Un ensemble* V_s *est un stopping-set s'il est un point fixe du décodeur à effacement.*

Cette définition peut être caractérisée par une équation [RW04]. Soit la fonction f qui compte le nombre de "1" sur un vecteur colonne v_c et définie par :

$$f(v_c) = \sum_j I(v_c[j]) = 1) \tag{3.1}$$

où $I(x)$ une fonction d'indication.

Caractérisation 3.1 *Un ensemble V_s forme un stopping-set si*

$$f\left(\Delta_{V_s}\right) = 0 \tag{3.2}$$

où Δ_{V_s} est la somme réelle des vecteurs colonnes correspondants aux noeuds de variable de V_s

Exemple 3.1 : *Sur le graphe représenté par la figure 3.1, l'ensemble $V_s = \{v_1, v_3, v_5\}$ défini un stopping-set dont les voisins sont c_1, c_2, c_4. En termes d'équation nous avons $\Delta_{V_s} = [2,\ 3,\ 3]^T$ donc $f\left(\Delta_{V_s}\right) = 0$*

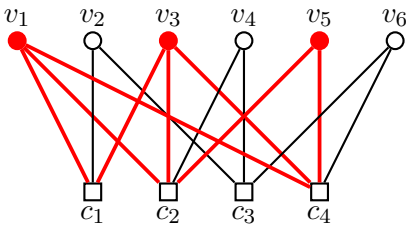


FIGURE 3.1 – Graphe de Tanner
dont $V_3 = \{v_1, v_3, v_5\}$ est stopping-set

Pour un canal binaire à effacement (CBE ou BEC en anglais), si tous les nœuds de variable d'un stopping-set sont effacés alors ces nœuds de variable ne pourront pas être reconstruits car chaque équation de parité du décodeur manque au moins deux valeurs or là la reconstruction n'est possible que s'il manque une valeur. Donc les stopping-sets, particulièrement ceux de petits tailles, vont influencer les corrections des décodeurs itératifs sur le canal à effacements [DPT+02].

Des méthodes de construction de code LDPC sans stopping-set de petites tailles ont été développées [Ric06, KHP11], dont la plupart sont pour des codes LDPC irréguliers et se basent sur les Approximate cycle EMD (ACE) [TJVW03, RW04, TJVW04, VS08]. En effet, à l'exception des codes réguliers en colonne de degré $d_v = 2$, où les plus petits stopping-sets sont composés d'un seul cycle (figure 3.2), les stopping-sets sont généralement constitués de plusieurs cycles. Donc éviter les stopping-sets de petites tailles revient en partie à éviter les combinaisons de plusieurs cycles courts formés par peu de nœuds de variable, c'est le concept de la connectivité des cycles et constitue l'idée central des algorithmes ACEs.

Les algorithmes ACEs permettent une construction colonne-par-colonne pour améliorer le plancher d'erreurs. Pour chaque nouvelle connexion l'algorithme cherche

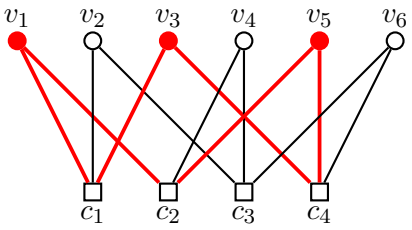


FIGURE 3.2 – Graphe de Tanner du code régulier $\mathcal{C}(6,2,3)$ de l'exemple 1.1, $S_3 = \{v_1, v_3, v_5\}$ est stopping-set et $(v_1, c_1, v_3, c_4, v_5, c_2, v_1)$ est un cycle

celle qui donne de plus longs cycles avec de large ACE [TJVW03, RW04, TJVW04, VS08]. Ce pendant ces algorithmes sont signifiants que pour les codes irréguliers. Pour comprendre, posons V_β un ensemble de β nœuds de variable, l'Approximate cycle EMD (ACE) est défini par :

$$ACE(V_\beta) = \sum_{i=1}^{\beta} (d_{v_{n_i}} - 2), \tag{3.3}$$

où $d_{v_{n_i}}$ est le degré du i^{eme} nœud de variable de V_β. L'$ACE(V_\beta)$ représente la valeur maximale du *Extrinsic Message Degree (EMD)*. En effet, l'EMD d'un ensemble de nœuds de variable V_β est le nombre de voisin de V_β qui ont une seule connexion avec lui. Ainsi pour un cycle de longueur 2β qui n'admet pas de sous-cycle, l'EMD de l'ensemble des nœuds de variable qui le forment est $\sum_{i=1}^{\beta}(d_{v_{n_i}} - 2)$ tandis que si le cycle admet des sous-cycles alors l'EMD est strictement inférieur à $\sum_{i=1}^{\beta}(d_{v_{n_i}} - 2)$. Si V_β est un stopping-set alors $EMD(V_\beta) = 0$. Pour les codes réguliers tous les ensembles de β nœuds de variable ont la même ACE qui vaut $\beta\,(d_v - 2)$ de ce fait l'ACE n'est signifiant que pour les codes irréguliers.

3.1.2 Les Trapping-Sets

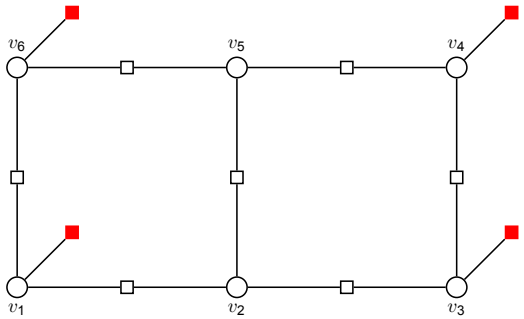
Les trapping-sets sont introduits par Richarson dans [Ric03], pour décrire la structure d'un ensemble $\mathbf{T}$ de nœuds de variable non corrigés à la fin des itérations. Pour les canaux BEC, ils sont appellés stopping-sets mais pour les canaux binaires symétrique (CBS ou BSC en anglais) et les canaux à bruit blanc additif gaussian (en anglais AWGN pour Additive White Gaussian Noise), c'est l'appelation trapping-set (TS) qui est utilisée.

Une notation commune utilisée pour décrire un TS est (a, b), où $a = |\mathbf{T}|$ est le nombre de nœuds de variable de l'ensemble $\mathbf{T}$, et b est le nombre de nœuds de contrôle de degré impair dans le sous-graphe formé par l'ensemble des nœuds de variable $\mathbf{T}$. La notation $\mathcal{T}(a, b)$ est utilisée pour dénoter le graphe biparti associé

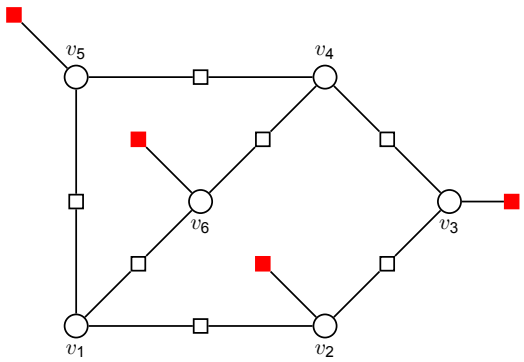
avec le trapping-set (a, b). Un graphe G contient un (a, b) de type $\mathcal{T}$ s'il existe un sous ensemble de nœuds de variable $\mathbf{T}$ dans G dont le sous graphe induit est isomorphe à $\mathcal{T}(a, b)$. Mais la notation (a, b) est insuffisante pour décrire la topologie d'un trapping-set particulier car il peut y avoir plusieurs graphes non-isomorphes avec a nœuds de variable et b nœuds de contrôle de degré impair. Par conséquent nous utilisons la notation qui énumère les différents cycles dans le sous graphe [DVPL13, KB14].

Définition 3.3 *Le trapping-set associé à $\mathcal{T}$ est dit de type $(a, b; \prod_{k\geq 2}(2k)^{g_{2k}})$ si $\mathcal{T}$ contient g_{2k} cycles de longueur $(2k)$, où $k \geq 2$.*

Par exemple sur la figure 3.3, nous avons 2 trapping-sets (6,4) qui sont différenciés par les cycles qui les composent. La figure 3.3(a) est un trapping-set de type $\left(6, 4; 8^2, 12^1\right)$ et la figure 3.3(b) est un trapping-set de type $\left(6, 4; 8^1, 10^2\right)$.



(a)



(b)

FIGURE 3.3 – Trapping-set (6,4)

Un TS est dit *élémentaire* si $\mathcal{T}$ contient seulement des nœuds de contrôle de degré un et/ou deux sinon le TS est non-élémentaire. Les trapping-sets élémentaires dénotés par TSEs sont connus comme étant les dominants dans le plancher

d'erreurs [Ric03, CSV06, NVMC10, NCMV12, KB14]. Ainsi, nous founissons dans cette thèse une méthode d'éviter de petits TSEs.

3.1.3 Absorbing-Sets (AS)

Soit G_{as} un sous-graphe de G constitué d'un sous ensemble $V_{as} \subset V$ de nœuds de variable et d'un sous ensemble $C_{as} \subset C$ de nœuds de contrôle. Notons $o(V_{as}) \subset C_{as}$ l'ensemble des voisins de V_{as} avec un degré impair dans V_{as} et $e(V_{as}) \subset C_{as}$ l'ensemble des voisins de V_{as} avec un degré pair dans V_{as}. Les absorbing-sets sont des cas particuliers de trapping-sets qui sont définis comme suit :

Définition 3.4 (cf. [DZA$^+$10]) : *Un absorbing-set (n_{as}, n_o), noté $AS(n_{as}, n_o)$, est un sous-ensemble $V_{as} \subset V$ de nœuds de variable avec $n_{as} = |V_{as}|$ et $n_o = |o(V_{as})|$, où chaque nœud de V_{as} a un nombre de voisins dans $o(V_{as})$ stictement inférieurs au nombre de voisins dans $e(V_{as})$. L'absorbing-set est dit* **fully absorbing-set**, *si chaque nœud de variable dans $V \setminus V_{as}$ a strictement plus de voisins dans $C \setminus o(V_{as})$ que dans $o(V_{as})$.*

D'après cette définition du absorbing-set, tout $AS(n_{as}, n_o)$ est un trapping-set (n_{as}, n_o). Une propriété importante des *fully absorbing-sets* est qu'ils sont "points fixes" pour un décodage bit-flipping dans lequel les valeurs des bits correspondants aux nœuds de variable sont changées si la majorité des fonctions de parités correspondantes aux nœuds de contrôle voisins ne sont pas satisfaites [DZA$^+$10].

La figure 3.4 montre un $AS(4, 8)$, qui a quatre nœuds de variable "◯" et huit nœuds de contrôle à degré impair "□". L'ajout d'un cinquième nœud de variable "•" et d'un sixième nœud de variable "•" comme sur les figures de 3.5 montre que les fully absorbing-sets contiennent plus de cycles courts, que les structures non-fully absorbing-sets. L'ajout du sixième nœud de variable donne un nouveau 6-cycle pour le cas non-fully (figure 3.5(a)) et cinq nouveaux cycles dans le cas fully (3.5(b)). Ces résultats montrent encore l'importance d'éviter des fully absorbing-sets et plusieurs travaux ont été effectués pour éviter les ASs faibles [DZA$^+$10, DWZ10, WDZW11, WDW13].

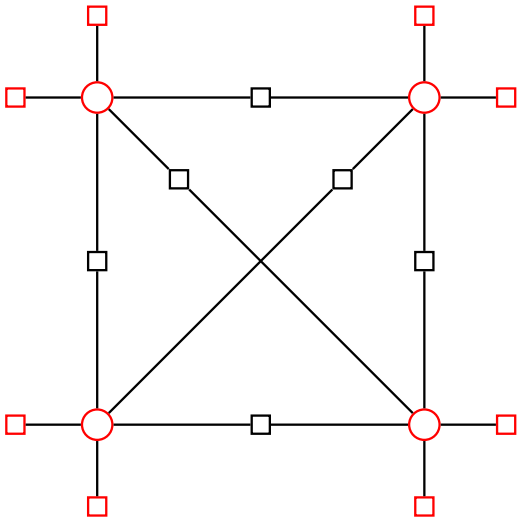
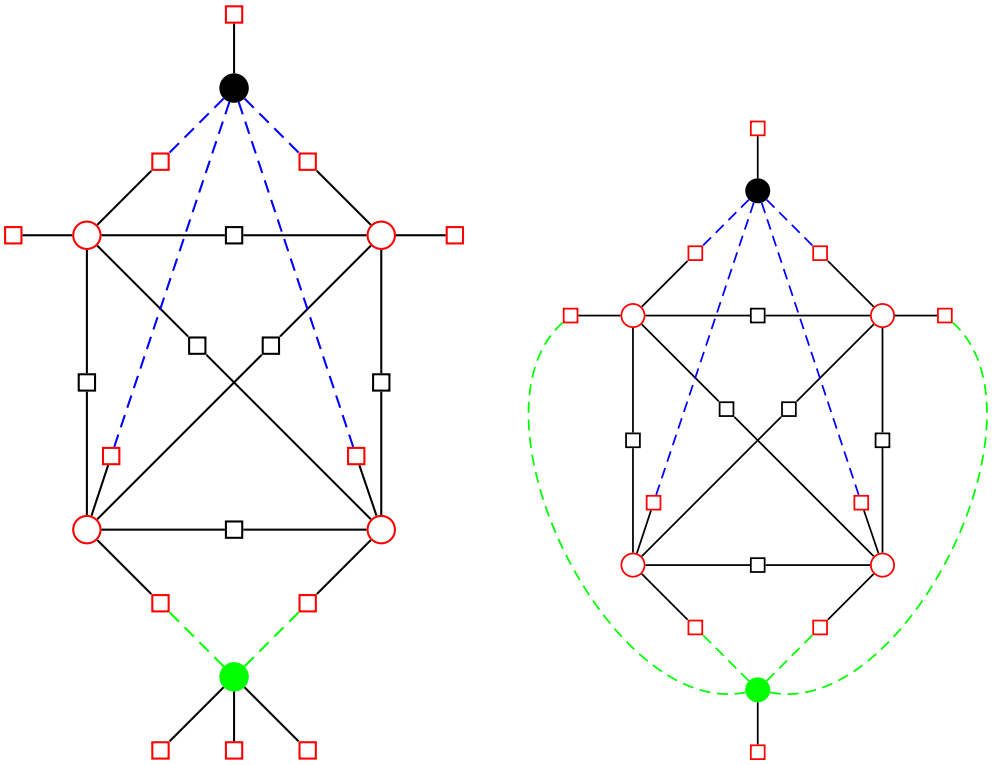


FIGURE 3.4 – Absorbing-set (4,8)



(a) (b)

FIGURE 3.5 – 3.5(a) Non fully absorbing-set (4,8),
3.5(b) fully absorbing-set (4,8)

3.1.4 Les Expansions

Les stopping-sets, les absorbing-sets et les trapping-sets peuvent tous être considérés comme un sous graphe de G avec **a** nœuds de variable connectés à **b** nœuds de parité. Pour une étude généralisée de ces topologies nous utilisons la notion d'*expansion-set* définie comme suit :

Définition 3.5 : *Une expansion-set $E(a,b)$ est un sous graphe de G formé par* **a** *nœuds de variable interconnectés avec* **b** *nœuds de contrôle. La notation $\mathcal{E}(a,b)$ est utilisée pour dénoter le graphe biparti associé avec l'expansion-set $E(a,b)$. L'expansion-set est dite élémentaire si chacun des* **b** *nœuds de contrôle a au plus deux voisins parmi les* **a** *nœuds de variable.*

Comme avec les trapping-sets, pour les expansion-sets il existe aussi plusieurs $E(a,b)$ non-isomorphes et pour les différencier nous intégrons les cycles qui les forment dans la notation. Ainsi l'écriture d'une expansion devient $\mathcal{E}(a,b;\prod_{k\geq 2}(2k)^{g_{2k}})$ où g_{2k} est le nombre de cycles de longueur $(2k)$ dans $\mathcal{E}$. Il existe une relation entre a et b en fonction du nombre de cycle qui forme l'expansion comme le montre les théorèmes suivants.

Théorème 3.1 : *Une expansion $E(a,b)$ est formé d'un simple cycle si*

$$b = a\,(d_v - 1)$$

Preuve : *Dans un cycle de longueur* **2a** *nous avons* **a** *nœuds de variable et* **a** *nœuds de contrôle, de plus chaque nœud de variable a $d_v - 2$ voisins qui ne sont pas dans le cycle, par suite nous obtenons*

$$\begin{aligned} b &= a\,(d_v - 2) + a \\ &= a\,(d_v - 1) \end{aligned}$$

A noter que pour un code de girth $g \leq 2a$, les expansion-sets de type $E(a, a\,(d_v - 1))$ ne peuvent pas être évitées car ils traduisent les cycles de longueurs $2a$. Ainsi les premiers types de structures pouvant être évités sont composés de trois cycles. Nous donnons dans le théorème suivant la relation entre a et b pour une expansion-set élémentaire formée de trois cycles.

Théorème 3.2 : *Une expansion-set élémentaire $E(a,b)$ est formée par trois cycles si*

$$b = a\,(d_v - 1) - 1$$

Preuve : *Pour une structure avec trois cycles, nous avons deux chemins dans l'arbre des voisins (cf. Lemme 3.2). Sur le premier chemin* $v_n, \cdots, c_{m_1}^{l_1}$ *nous avons* l_1+1 *nœuds de variable et chaque nœud a* d_v-1 *fils excepté la racine qui a* d_v-2 *fils donc nous avons* $(l_1+1)(d_v-1)-1$ *nœuds de contrôle voisins sur le chemin* $v_n, \cdots, c_{m_1}^{l_1}$. *Sur le second chemin* $v_n, \cdots, c_{m_2}^{l_2}$ *nous avons* l_2+1 *nœuds de variable et si* n *est le nombre de nœuds de variable en commun sur les deux chemins alors nous avons* l_2+1-n *nouveaux nœuds de variable avec* $(l_2+1-n)(d_v-1)$ *nœuds de contrôle. Ainsi sur les deux chemins nous avons* $a = l_1+l_2+2-n$ *nœuds de variable et* $b = (l_1+l_2+2-n)(d_v-1)-1 = a(d_v-1)-1$.

Toutes ces topologies décrites ont des liens entre elles. Un trapping-set élémentaire $(a,0)$ est un stopping-set et l'expansion-set $E\left(a,b;\prod_{k\geq 2}(2k)^{g_{2k}}\right)$ correspond au trapping-set $\left(a,b-b_e;\prod_{k\geq 2}(2k)^{g_{2k}}\right)$ où b_e est le nombre de noeuds de contrôle de degré pair dans l'expansion-set. Par exemple, le $AS\,(4,8)$ de la figure 3.4 correspond au trapping-set $\left(4,8;6^4,8^3\right)$ qui est aussi l'expansion-set $E\left(4,14;6^4,8^3\right)$

Ces topologies étant nocives pour les décodeurs itératifs, il nous faut les caractériser pour pouvoir les détecter et les éviter lors de la conception.

3.2 Caractérisation et Critère de détection des TS$(5,3;8^3)$, TS$(6,4;8^2,12)$, et TS$(6,4;8,10^2)$

Les trapping-sets élémentaires sont connus comme étant dominants dans le plancher d'erreurs [Ric03, KB14] et parmi eux les plus nocifs sont les trapping-sets avec a et b faibles. Pour mieux comprendre cette affirmation nous définissons la relation topologique entre les différents trapping-sets.

Définition 3.6 (cf. [NCMV12]) : *Un trapping-set* $\mathcal{T}_2$ *est un successeur d'un trapping-set* $\mathcal{T}_1$ *s'il existe un sous ensemble de nœuds de variable de* $\mathcal{T}_2$ *qui induit un sous graphe isomorphe au sous graphe induit par* $\mathcal{T}_1$. *Si* $\mathcal{T}_2$ *est successeur de* $\mathcal{T}_1$ *alors* $\mathcal{T}_1$ *est prédécesseur de* $\mathcal{T}_2$. $\mathcal{T}_2$ *est un successeur direct de* $\mathcal{T}_1$ *si* $\mathcal{T}_2$ *n'admet pas un prédécesseur* $\mathcal{T}_3$ *lequel est successeur de* $\mathcal{T}_1$.

Corollaire 3.1 : *Soit* $\mathcal{T}_2(a_2,b_2)$ *un successeur de* $\mathcal{T}_1(a_1,b_1)$, *si* $a_1 < a_2$ *alors l'existence de* $\mathcal{T}_1$ *est une condition nécessaire de l'existence de* $\mathcal{T}_2$.

Ce corollaire montre l'impact des petits trapping-sets. Une conséquence est : *si un trapping-set* $\mathcal{T}_1$ *n'existe pas dans le graphe d'un code alors tous ses successeurs n'existe pas dans le graphe.* Ainsi l'élimination des petits trapping-sets entraine l'élimination de tous les trapping-sets successeurs de ces petits trapping-sets. Il

est donc important de connaître la hiérarchie des TSEs pour un code régulier de degré d_v et de girth g_t. Dans un premier temps nous nous interessons sur les codes réguliers de degré $d_v = 3$ et de girth $g_t = 8$ dénoté $\mathcal{C}_{3,8}$.

La figure 3.6 tirée de [NCMV12] représente la hiérarchie des TSEs avec $4 \leq a \leq 8$, pour les codes $\mathcal{C}_{3,8}$. Une liste plus complète est dans [TSO].

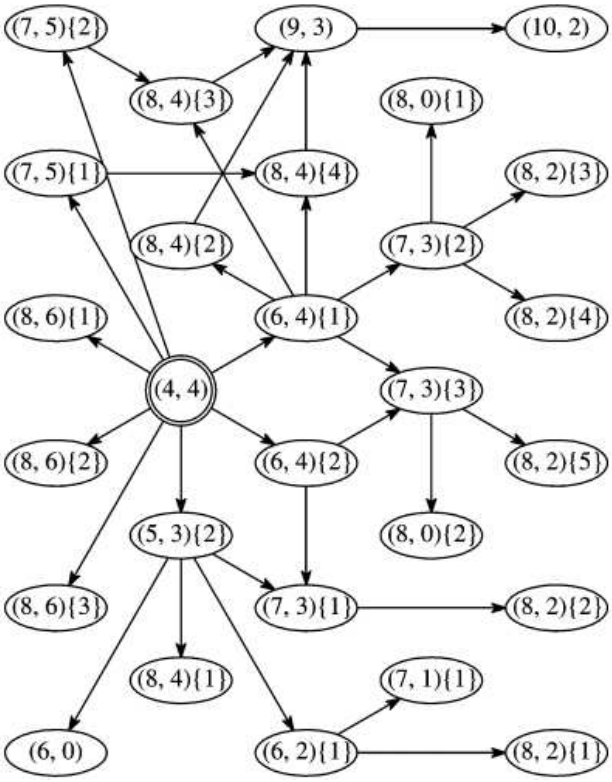


FIGURE 3.6 – Hiérachie des trapping-sets pour code régulier en colonne $d_v = 3$ et de girth $g_t = 8$

Sur cette représentation, nous remarquons que tous les TSEs avec b faible ($b \leq 4$ ou $a - b \leq 4$) sont des successeurs des trapping-sets $(5, 3)\,\{2\}$, $(6, 4)\,\{1\}$ et $(6, 4)\,\{2\}$ qui correspondent respectivement aux TSEs $(5, 3; 8^3)$, $(6, 4; 8^2, 12^1)$ et $(6, 4; 8^1, 10^2)$ par rapport à notre notation. Pour un code $\mathcal{C}_{3,8}$, les cycles de longueurs 8 correspondent aux trapping-sets $(4, 4; 8^1)$ donc traiter ces derniers revient à traiter les 8-cycles qui ne peuvent pas être éliminés mais seulement minimisés. Par suite les TSEs les plus nocifs pour un code $\mathcal{C}_{3,8}$ sont les TSEs $(5, 3; 8^3)$, $(6, 4; 8^2, 12^1)$ et $(6, 4; 8^1, 10^2)$, nous proposons une méthode prédictive pour détecter ces trapping-sets dans un arbre de voisins de profondeur-k. Avant cela, nous donnons les caractéristiques de ces trapping-sets.

3.2.1 Caractérisation

Les TSs pouvant être considérés comme une union de plusieurs cycles avec des nœuds de variable en commun. Nous caractérisons les TSEs $(5,3)$ et $(6,4)$ en fonction des cycles qui les forment et des nœuds de variable en commun. Ces caractérisations sont utilisées pour définir les critères de détection de ces trapping-sets.

Dans les représentations graphiques des TSEs ci-dessous, les nœuds de variable sont représentés par des cercles "◯", les nœuds de contrôle de degré impair par des carrés pleins rouges ■ et les nœuds de contrôle de degré pair par des carrés simples □.

Caractérisation 3.2 : *Un trapping-set élémentaire* $(5,3;8^3)$ *peut être caractérisé comme une union de deux 8-cycles ayant exactement trois nœuds de variable en commun.*

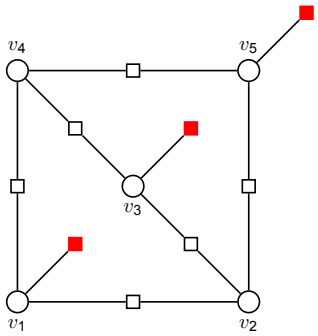


FIGURE 3.7 – Représentation graphique du TS$(5,3;8^3)$

La figure 3.7 représente un trapping-set $(5,3;8^3)$, où les ensembles des nœuds de variable des trois cycles sont $\{v_1,v_2,v_3,v_4\}$, $\{v_2,v_3,v_4,v_5\}$, et $\{v_1,v_2,v_4,v_5\}$, ces ensembles pris deux à deux ont trois nœuds de variable en commun.

Lemme 3.1 : *Il n'existe pas de trapping-set non-élémentaire de type* $\mathcal{T}\left(5,3;8^3\right)$.

Preuve : *Soit un* $\mathcal{C}_{3,8}$*, un 8-cycle est composé de 4 nœuds de variable avec 8 nœuds de contrôle voisins dont 4 sont de degré 1 et les 4 autres sont de degré deux. L'addition d'un nouveau nœud de variable qui donne un nouveau 8-cycle implique que 2 nœuds de contrôle parmi les 4 de degré 1 forment les branches avec le nouveau nœud de variable. Donc les* $\mathcal{T}(5,3)$ *dans un graphe d'un code* $\mathcal{C}_{3,8}$ *contiennent uniquement des nœuds de contrôle de degré 1 et/ou 2.*

Caractérisation 3.3 : *Un trapping-set élémentaire de type* $(6,4;8^2,12^1)$ *peut être caractérisé comme une union de deux 8-cycles ayant exactement deux nœuds de variable en commun.*

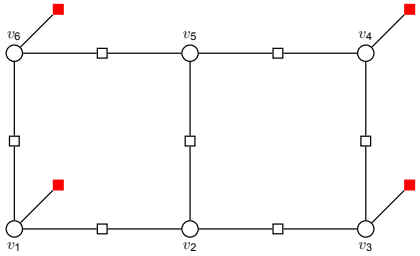


FIGURE 3.8 – Représentation graphique du TS$(6, 4; 8^2, 12^1)$

La figure 3.8 montre un $\mathcal{T}\left(6, 4; 8^2, 12^1\right)$ où nous avons deux 8-cycles dont les ensembles des nœuds de variable sont $\{v_1, v_2, v_5, v_6\}$ et $\{v_2, v_3, v_4, v_5\}$, ces deux ensembles ont v_2 et v_5 en commun.

Caractérisation 3.4 : *Un trapping-set élémentaire de type* $(6, 4; 8^1, 10^2)$ *peut être caractérisé comme une union de deux 10-cycles ayant exactement quatre nœuds de variable en commun.*

Bien que le 8-cycle existe dans le graphe, il n'est pas utilisé pour caractériser le trapping-set. La figure 3.9 représente un $\mathcal{T}\left(6, 4; 8^1, 10^2\right)$ dont $\{v_1, v_2, v_3, v_4,$ $v_5\}$ et $\{v_1, v_2, v_3, v_4, v_6\}$ sont les ensembles des nœuds de variable des deux 10-cycles et $\{v_1, v_2, v_3, v_4\}$ sont les nœuds de variable en commun.

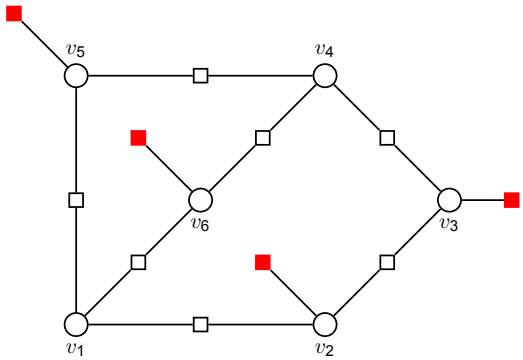


FIGURE 3.9 – Représentation graphique du TS$(6, 4; 8^1, 10^2)$

3.2.2 Critère de détection

L'algorithme PEG permet de détecter de manière prédictive les cycles selon le niveau d'apparition de chaque nœud de contrôle dans l'arbre de profondeur-k. Ce pendant la détection des structures complexes tel que les TS est plus délicate et il n'existe pas encore de méthode basée sur l'algorithme PEG pour détecter et éviter un TS particulier de manière prédictive.

Khazraie et al. ont proposé dans [KAB12] une méthode de construction basée sur le PEG mais cette méthode n'est prédictive. Elle consiste à établir une connexion avec $v_{l.p}$ pour chaque nœud de contrôle de l'ensemble des potentiels candidats puis d'effectuer un test de vérification avec l'algorithme de Karimi et al. [KB12]. Si la connexion donne un TSE alors ce nœud de contrôle est supprimé des potentiels candidats. Le test de vérification montre que la méthode n'est pas prédictive et ce test augmente la complexité lors de la conception.

Dans l'arbre de profondeur-k, un TS peut être prédit en utilisant le nombre de copies d'un nœud de contrôle avec leurs niveaux d'apparition et le nombre de nœuds de variable en commun sur les différents chemins entre la racine et chaque copie du nœud de contrôle. Nous rappelons et introduisons quelques notations et définitions relatives à l'arbre des voisins. L'arbre des voisins de racine v_n est dénoté par T_{v_n}. Lors de l'expansion de l'arbre de profondeur-k, les nœuds de variable sont sur les niveaux pair tandis que les nœuds de contrôle sur les niveaux impairs, et différentes copies d'un même nœud de variable et/ou contrôle peuvent apparaître à différents niveaux sur T_{v_n}. Pour différencier ces copies nous utilisons la notation $u_i^{l_i}$ où i dénote le i^{eme} copie du nœud u et l_i son niveau d'apparition.

Définition 3.7 : *Un nœud $w \in T_{v_n}$ est dit un* **ascendant** *du nœud $u \in T_{v_n}$ s'il existe un chemin de la racine v_n au nœud u qui traverse le nœud w. L'ensemble des nœuds de variable ascendants du nœud u dans T_{v_n} est noté $\mathcal{A}(u)$. Un nœud $w \in T_v$ est dit un* **parent** *du nœud $u \in T_v$ si w est directement connecté à u sur le chemin traversant depuis la racine v_n. Pour un ensemble U donné, $\mathcal{A}_{T_{v_n}}(U)$ dénote l'ensemble des nœuds de variable ascendants de tout $u \in U$.*

Définition 3.8 : *Un nœud $w \in T_{v_n}$ est dit un* **descendant** *du nœud $u \in T_v(G)$ s'il existe un chemin de la racine v_n au nœud w qui traverse le nœud u. L'ensemble des nœuds de variable descendants du nœud u dans T_{v_n} est noté $\mathcal{D}(u)$. Un nœud $w \in T_v$ est dit un* **fils** *du nœud $u \in T_v$ si u est directement connecté à w sur le chemin traversant depuis la racine v_n. Pour un ensemble U donné, $\mathcal{D}_{T_{v_n}}(U)$ dénote l'ensemble des nœuds de variable descendants de tout $u \in U$. Un nœud qui n'a pas de fils est une feuille.*

Les TSEs étant composés de plusieurs cycles et l'apparition de plusieurs copies d'un nœud sur l'arbre des voisins prouve l'existence de cycle, il nous faut savoir le nombre de copies à traiter suivant le nombre de cycle voulu. Dans les théorèmes suivants nous donnons le nombre maximum de cycles qui peut être créé par une nouvelle connexion entre le nœud de variable v_n et le nœud de contrôle c_m qui a β copies. Par la suite, nous donnons le nombre de copies nécessaire à considérer pour une union de α cycles.

Théorème 3.3 : *Soit T_{v_n} contenant β copies du nœud de contrôle c_m, le nombre maximum de cycles déjà présent dans T_{v_n} et contenant c_m est $\eta = C_2^{\beta} = \binom{\beta}{2}$. Si nous choisissons de connecter v_n et c_m pour la nouvelle branche alors le nombre maximum de cycles pour la nouvelle structure est $\eta + \beta = \frac{\beta(\beta+1)}{2}$.*

Preuve : *Soit T_{v_n} contenant β copies du nœud de contrôle c_m, le nombre maximum de cycles déjà présent dans T_{v_n} et contenant c_m est le nombre de combinaisons deux chemins, c'est $\eta = C_2^{\beta} = \frac{\beta(\beta-1)}{2}$, si de plus nous connectons v_n et c_m nous avons β nouveaux cycles par suite nous obtenons au total $\eta+\beta = \frac{\beta(\beta-1)}{2}+\beta = \frac{\beta(\beta+1)}{2}$ cyles.*

Les trapping-sets élémentaires $(5,3)$ et $(6,4)$ pour un code $\mathcal{C}_{3,8}$, sont formé par trois cycles. Il est essentiel de donner le nombre de copies requis du nœud de contrôle pour former une combinaison de α cycles.

Lemme 3.2 : *Une nouvelle combinaison de α cycles est détecté dans T_{v_n}, si le nœud de contrôle sélectionné a au moins β copies avec $\beta = \lfloor \frac{-1+\sqrt{1+8\alpha}}{2} \rfloor$.*

Preuve : *Soit β le nombre de copie nécessaire pour avoir au maximum α cycles donc nous avons la relation suivante :*

$$\begin{aligned} \frac{\beta(\beta+1)}{2} \leq \alpha \;&\Rightarrow\; \frac{\beta(\beta+1)}{2} - \alpha \leq 0 \\ &\Rightarrow\; \beta^2 + \beta - 2\alpha \leq 0 \end{aligned}$$

$$\Delta = 1 + 8\alpha$$

$$\beta_1 = \frac{-1-\sqrt{1+8\alpha}}{2} \quad \beta_2 = \frac{-1+\sqrt{1+8\alpha}}{2}$$

La solution de l'inéquation $\frac{\beta(\beta+1)}{2} \leq \alpha$ est alors l'intervalle $\left[\frac{-1-\sqrt{1+8\alpha}}{2}; \frac{-1+\sqrt{1+8\alpha}}{2}\right]$ et la valeur entière maximale $\left\lfloor \beta = \frac{-1+\sqrt{1+8\alpha}}{2} \right\rfloor$

Une conséquence du lemme 3.2 est qu' une nouvelle combinaison de trois cycles est détectée si le nœud de contrôle c_m à deux copies. Il suffit de remplacer α par trois dans le lemme, nous obtenons $\beta = 2$. Avec les deux copies nous avons deux nouveaux cycles de longueurs respectives $l_1 + 1$ et $l_2 + 1$ où l_i est le niveau d'apparition de la i^{eme} copie avec $1 \leq i \leq 2$. Il reste à évaluer la longueur du cycle qui existe déjà car tout chemin de $c_{m_1}^{l_1}$ à $c_{m_2}^{l_2}$ forme un cycle. Le théorème suivant donne la longueur d'un cycle existant à partir de la profondeur des deux copies $c_{m_1}^{l_1}$ et $c_{m_2}^{l_2}$.

Théorème 3.4 : *Soit v_n un nœud de variable d'un trapping-set élémentaire et c_m un nœud de contrôle dans T_{v_n} avec β copies dénotées $c_{m_i}^{l_i}$, $1 \leq i \leq \beta$. La longueur du cycle formé par les deux chemins distincts $v_n, \cdots, c_{m_1}^{l_1}$ et $v_n, \cdots, c_{m_2}^{l_2}$ est définie comme suit :*

1. *$g = l_1 + l_2 + 6 - 4n$, si $c_{m_1}^{l_1}$ et $c_{m_2}^{l_2}$ ont même parent.*
2. *$g = l_1 + l_2 + 4 - 4n$, si $c_{m_1}^{l_1}$ et $c_{m_2}^{l_2}$ n'ont pas même parent.*

où n est le nombre de nœuds de variable en commun sur les deux chemins (ie. $n = \left|\mathcal{A}(c_{m_1}^{l_1}) \bigcap \mathcal{A}(c_{m_2}^{l_2})\right|$).

Preuve : *Supposons qu'il existe dans T_{v_n}, β copies du nœud de contrôle c_m. Soient deux copies de c_m dénotées par $c_{m_1}^{l_1}$ à $c_{m_2}^{l_2}$ dont les ensembles des ascendants sont respectivement $\mathcal{A}(c_{m_1}^{l_1})$ et $\mathcal{A}(c_{m_2}^{l_2}$. En posant $n = \left|\mathcal{A}(c_{m_1}^{l_1}) \bigcap \mathcal{A}(c_{m_2}^{l_2})\right|$, le nombre de nœuds de variable dans $\mathcal{A}(c_{m_1}^{l_1}) \bigcup \mathcal{A}(c_{m_2}^{l_2})$ est :*

$$\begin{aligned}\left|\mathcal{A}(c_{m_1}^{l_1}) \bigcup \mathcal{A}(c_{m_2}^{l_2})\right| &= \frac{l_1+1}{2} + \frac{l_2+1}{2} - n \\ &= \frac{l_1 + l_2 + 2 - 2n}{2}\end{aligned}$$

- *Si $c_{m_1}^{l_1}$ et $c_{m_2}^{l_2}$ ont le même parent alors il existe deux nœuds de variable parmi les nœuds en commun qui appartiennent à l'ensemble des nœuds de variable qui forment le cycle donc nous avons $\frac{l_1+l_2+2-2n}{2} - (n-2)$ nœuds de variable dans le cycle ce qui donne un cycle de longueur :*

$$\begin{aligned}g &= 2\frac{l_1 + l_2 + 6 - 4n}{2} \\ &= l_1 + l_2 + 6 - 4n\end{aligned}$$

- *Si $c_{m_1}^{l_1}$ et $c_{m_2}^{l_2}$ n'ont pas le même parent alors il existe un nœud de variable parmi les nœuds en commun qui appartiennent à l'ensemble des nœuds de variable qui forment le cycle donc nous avons $\frac{l_1+l_2+2-2n}{2} - (n-1)$ nœuds de variable dans le cycle ce qui donne un cycle de longueur :*

$$\begin{aligned}g &= 2\frac{l_1 + l_2 + 4 - 4n}{2} \\ &= l_1 + l_2 + 4 - 4n\end{aligned}$$

Après ces théorèmes et lemmes liés aux copies d'un nœud de contrôle, nous pouvons décrire maintenant comment détecter un TSE (5,3) et (6,4) dans T_{v_n}. Le théorème suivant est le premier théorème majeur qui définit comment détecter un trapping-set $(5, 3; 8^3)$ dans un arbre de voisins.

Théorème 3.5 : *Soit un code $\mathcal{C}_{3,8}$, pour tout nœud de variable v_n un trapping-set $(5,3;8^3)$ est détecté dans T_{v_n} si et seulement si il existe au moins 2 copies du nœud de contrôle c_m dénotés par $c^7_{m_1}$, $c^7_{m_2}$ qui ont le même parent, et pour lesquels le chemin $v_n,\cdots,c^7_{m_1}$ et le chemin $v_n,\cdots,c^7_{m_2}$ dans T_{v_n} ont 3 nœuds de variable en commun (ie. $\left|\mathcal{A}(c^7_{m_1})\bigcap\mathcal{A}(c^7_{m_2})\right|=3$).*

Preuve : *Un TSE $(5,3;8^3)$ est constitué de trois cycles. Soit un code $\mathcal{C}_{3,8}$, et T_{v_n} qui contient β d'un nœud de contrôle c_m tel que deux copies soient dénotées par $c^{l_1}_{m_1}$, $c^{l_2}_{m_2}$. Pour un TSE tous les nœuds de contrôle ont au maximum deux voisins, donc si $c^{l_1}_{m_1}$, $c^{l_2}_{m_2}$ n'ont pas le même parent et qu'ils soient connectés à v_n alors le trapping-set n'est pas élémentaire.*

Soit $g_1=l_1+1$ et $g_2=l_2+1$ et $n=\left|\mathcal{A}(c^7_{m_1})\bigcap\mathcal{A}(c^7_{m_2})\right|$.

- *si $l_1\neq 7$ (resp. $l_2\neq 7$) alors $g_1\neq 8$ (resp. $g_2\neq 8$). Il existe des cycles de longueurs différentes de 8, or $(5,3;8^3)$ est composé que de 8-cycle par suite il n'ya pas de TSE $(5,3;8^3)$.*
- *Pour $l_1=7$ et $l_2=7$, la longueur du cycle formé par les deux chemins distincts $v_n,\cdots,c^7_{m_1}$ et $v_n,\cdots,c^7_{m_2}$ est $g=7+7+6-4n$ (cf. théorème 3.4). Finalement $g=20-4n$, si $n\neq 3$ alors $g\neq 8$ par contre si $n=3$ alors $g=8$ et $\left|\mathcal{A}(c^7_{m_1})\bigcup\mathcal{A}(c^7_{m_2})\right|=5$. Par conséquent nous avons un TSE $(5,3;8^3)$, si $l_1=l_2=7$ et $n=3$.*

Les figures 3.10 et 3.11 montrent les deux scénarios pouvant conduire à un TSE $(5,3;8^3)$ à partir d'un 8-cycle. Sur la figure 3.10 le nœud de variable v_5 est déjà connecté sur un nœud de contrôle de degré impair d'un TS $(4,4;8^1)$ et il faut éviter que v_5 ait un autre voisin dans le trapping-set $(4,4;8^1)$ dont l'ensembe des nœuds de variable est $V_1=\{v_1,v_2,v_3,v_4\}$. Sur l'arbre des voisins les trois nœuds de variable en commun des deux chemins sont v_5,v_4,v_2 représenté en bleu et v_2 est le nœud parent de $c^7_{m_1}$ et de $c^7_{m_2}$. Par contre sur la figure 3.11 le nœud de variable v_5 est présent dans TS $(4,4;8^1)$ qu'il forme avec les nœuds de variable v_1,v_2,v_3. Sur l'arbre les nœuds en commun sur les deux chemins sont v_5,v_2,v_4 représentés en bleu. Les pointillés en vert représentent les connexions à éviter pour v_5.

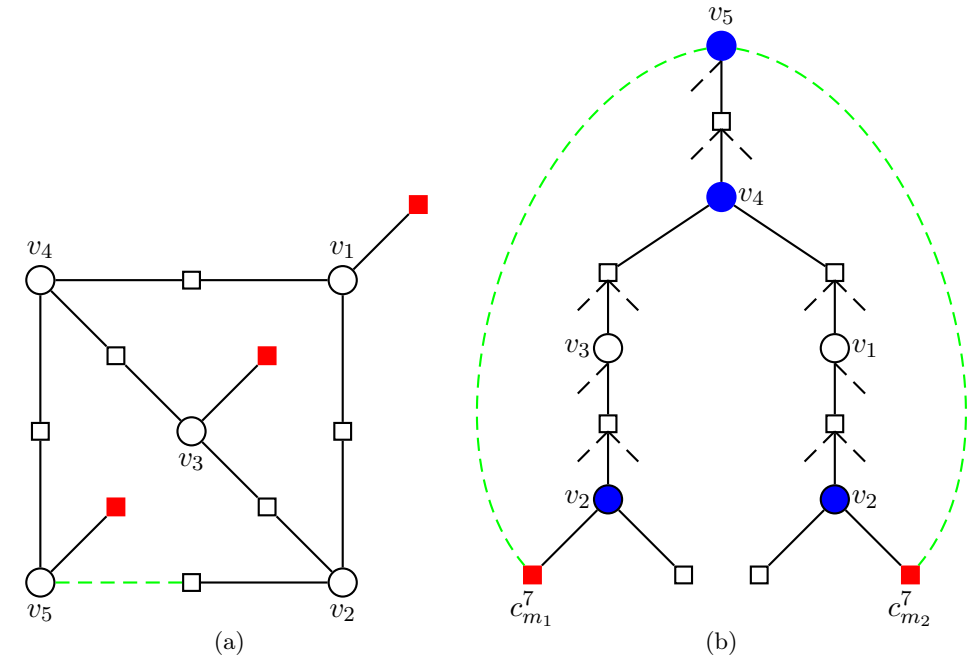


FIGURE 3.10 – Premier scenario pour éviter un $(5, 3; 8^3)$

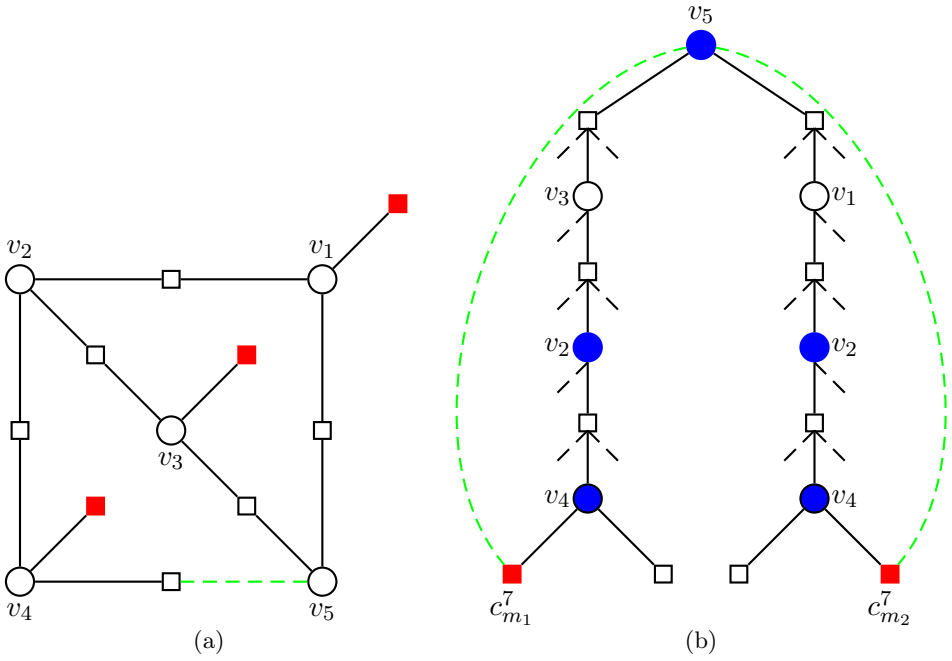
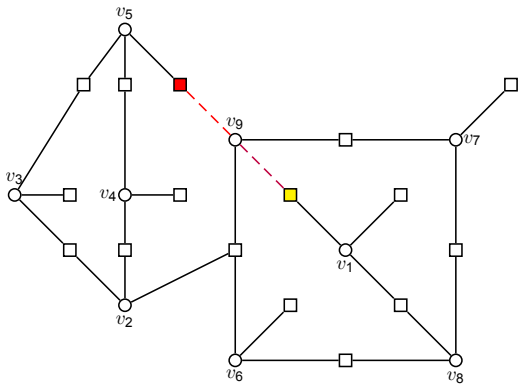


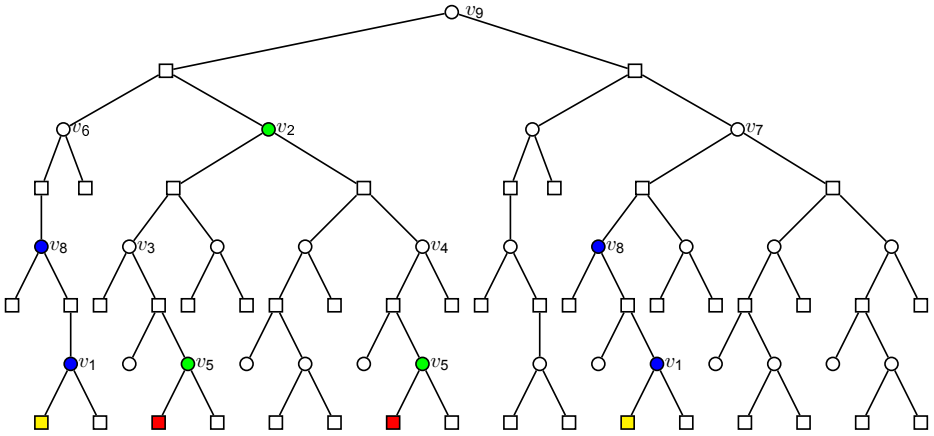
FIGURE 3.11 – Deuxième scenario pour éviter un $(5, 3; 8^3)$

Il faut noter que sur un arbre les deux scénarios peuvent y être, la figure 3.12 en est une parfaite illustration où les connexions à éviter pour v_9 sont en pontillés sur la figure 3.12(a), les chemins de nœuds de variable $\{v_2, v_3, v_5, v_9\}$ et $\{v_2, v_4, v_5, v_9\}$

représente le premier scenario et les chemins de nœuds de variable $\{v_1, v_6, v_8, v_9\}$ et $\{v_1, v_7, v_8, v_9\}$ représente le deuxième scénario. Sur l'arbre les nœuds de variable en commun sont les cercles pleins, en vert pour le premier scénario et en bleu pour le deuxème scénario, les nœuds de contrôle à éviter sont aussi en carré plein colorié.



(a)



(b)

FIGURE 3.12 – Les deux scénarios présentés sur le même graphe

De plus les nœuds de contrôle à éviter peuvent avoir plus de deux copies avec les critères de TSE $(5, 3; 8^3)$ dans l'arbre. Comme le montre la figure 3.13 où le nœud de contrôle en rouge connecté avec v_1 et v_5 est à éviter.

Après le théorème sur la détection d'un TSE $\left(5, 3; 8^3\right)$, nous donnons les deux théorèmes pour la détection des TSEs $(6, 4)$.

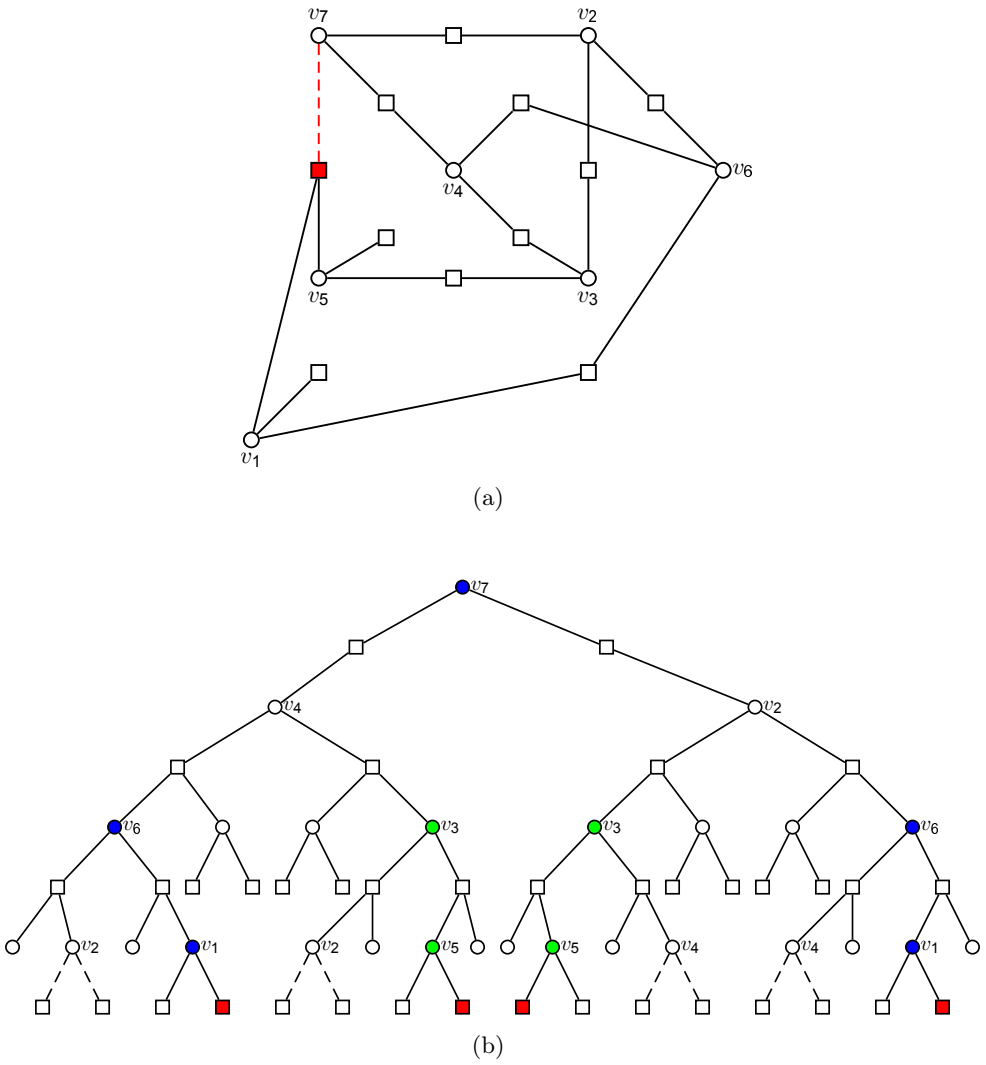


FIGURE 3.13 – Nœuds à éviter a plus de deux copies sur l'arbre

Théorème 3.6 : *Soit un code $\mathcal{C}_{3,8}$, pour tout nœud de variable v_n un trapping-set $(6,4;8^2,12^1)$ est détecté dans T_{v_n} si et seulement si l'un des cas suivants est vérifié :*

- 1^{er} **Cas :** *il existe au moins 2 copies du nœud de contrôle c_m dénotés par $c^7_{m_1}$, $c^{11}_{m_2}$ qui ont le même parent, et pour lesquels le chemin $v_n, \cdots, c^7_{m_1}$ et le chemin $v_n, \cdots, c^{11}_{m_2}$ dans T_{v_n} ont 4 nœuds de variable en commun (ie. $\left|\mathcal{A}(c^7_{m_1}) \bigcap \mathcal{A}(c^{11}_{m_2})\right| = 4$).*
- 2^{eme} **Cas :** *il existe au moins 2 copies du nœud de contrôle c_m dénotés par $c^7_{m_1}$, $c^7_{m_2}$ qui ont le même parent, et pour lesquels le chemin $v_n, \cdots, c^7_{m_1}$ et le chemin $v_n, \cdots, c^7_{m_2}$ dans T_{v_n} ont 2 nœuds de variable en commun (ie. $\left|\mathcal{A}(c^7_{m_1}) \bigcap \mathcal{A}(c^7_{m_2})\right| = 2$).*

Preuve : *Un TSE* $(6,4;8^2,12^1)$ *est constitué de trois cycles. Soit un code* $\mathcal{C}_{3,8}$*, et* T_{v_n} *qui contient* β *copies d'un nœud de contrôle* c_m *tel que deux copies soient dénotées par* $c_{m_1}^{l_1}$, $c_{m_2}^{l_2}$*. Pour un TSE tous les nœuds de contrôle ont au maximum deux voisins, donc si* $c_{m_1}^{l_1}$, $c_{m_2}^{l_2}$ *n'ont pas le même parent et qu'ils soient connectés à* v_n *alors le trapping-set n'est pas élémentaire.*

Soit $g_1 = l_1 + 1$ *et* $g_2 = l_2 + 1$ *et* $n = \left|\mathcal{A}(c_{m_1}^7) \bigcap \mathcal{A}(c_{m_2}^7)\right|$*. Un TS*$(6,4;8^2,12)$ *contient deux 8-cycles et un 12-cycle donc il n'y a que deux cas de niveau pour* c_m

- 1^{er} **cas :** *(*$l_1 = 7$,$l_2 = 11 \Rightarrow$ $g_1 = 8$,$g_2 = 12$ *ou (*$l_1 = 11$,$l_2 = 7 \Rightarrow$ $g_1 = 12$,$g_2 = 8$*, la longueur du cycle formé par les deux chemins distincts* $v_n, \cdots, c_{m_1}^{l_1}$ *et* $v_n, \cdots, c_{m_2}^{l_2}$ *est* $g = 7 + 11 + 6 - 4n$ *(cf. théorème 3.4). Finalement* $g = 24 - 4n$*, si* $n \neq 4$ *alors* $g \neq 8$ *par contre si* $n = 4$ *alors* $g = 8$ *et* $\left|\mathcal{A}(c_{m_1}^{l_1}) \bigcup \mathcal{A}(c_{m_2}^{l_2})\right| = 6$.
- 2^{eme} **cas :** *(*$l_1 = l_2 = 7 \Rightarrow$ $g_1 = g_2 = 8$*, la longueur du cycle formé par les deux chemins distincts* $v_n, \cdots, c_{m_1}^{l_1}$ *et* $v_n, \cdots, c_{m_2}^{l_2}$ *est* $g = 7 + 7 + 6 - 4n$ *(cf. théorème 3.4). Finalement* $g = 20 - 4n$*, si* $n \neq 2$ *alors* $g \neq 12$ *par contre si* $n = 2$ *alors* $g = 12$ *et* $\left|\mathcal{A}(c_{m_1}^{l_1}) \bigcup \mathcal{A}(c_{m_2}^{l_2})\right| = 6$.

Par suite, les seuls cas où nous avons un TSE $(6,4;8^2,12^1)$ *sont les cas où* $\left|\mathcal{A}(c_{m_1}^7) \bigcap \mathcal{A}(c_{m_2}^{11})\right| = 4$ *ou* $\left|\mathcal{A}(c_{m_1}^7) \bigcap \mathcal{A}(c_{m_2}^7)\right| = 2$.

Théorème 3.7 : *Soit un code* $\mathcal{C}_{3,8}$*, pour tout nœud de variable* v_n *un trapping-set* $(6,4;8^1,10^2)$ *est détecté dans* T_{v_n} *si et seulement si l'un des cas suivants est vérifié :*

- 1^{er} **Cas :** *il existe au moins 2 copies du nœud de contrôle* c_m *dénotés par* $c_{m_1}^7$, $c_{m_2}^9$ *qui ont le même parent, et pour lesquels le chemin* $v_n, \cdots, c_{m_1}^7$ *et le chemin* $v_n, \cdots, c_{m_2}^9$ *dans* T_{v_n} *ont 3 nœuds de variable en commun (ie.* $\left|\mathcal{A}(c_{m_1}^7) \bigcap \mathcal{A}(c_{m_2}^9)\right| = 3$*).*
- 2^{eme} **Cas :** *il existe au moins 2 copies du nœud de contrôle* c_m *dénotés par* $c_{m_1}^9$, $c_{m_2}^9$ *qui ont le même parent, et pour lesquels le chemin* $v_n, \cdots, c_{m_1}^9$ *et le chemin* $v_n, \cdots, c_{m_2}^9$ *dans* T_{v_n} *ont 4 nœuds de variable en commun (ie.* $\left|\mathcal{A}(c_{m_1}^9) \bigcap \mathcal{A}(c_{m_2}^9)\right| = 4$*).*

Preuve : *Preuve analogue à celle du théorème 3.6*

Dans cette partie nous avons caractérisé les trapping-sets élémentaires $(5,3)$ et $(6,4)$ à partir des cycles qui les forment. Puis nous avons donné les théorèmes qui définissent les relations entre le nombre de cycles d'une topologie et le nombre de copies nécessaires dans un arbre de calcul pour sa détection. En dernier point, nous avons défini des critères pour la détection prédictive des TSEs $(5,3)$ et $(6,4)$ dans un arbre. Ces critères de détection prédictive n'existent pas encore dans la

littérature. L'ajout de ces critères de détection sur la fonction coût de l'algorithme RandPEG permet de minimiser voir éliminer les TSEs (5,3) et (6,4) d'un code $\mathcal{C}_{3,8}$ avec une complexité réduite par rapport à la méthode non prédictive proposé par Khazraie et al. dans [KAB12].

3.3 Un algorithme RandPEG supprimant les TS : RandPEGnoTS

3.3.1 Présentation générale

L'algorithme RandPEG original permet uniquement de minimiser les cycles avec sa fonction coût. Nous créons une fonction coût qui évalue le nombre de trapping-sets $(5, 3)$ et $(6, 4)$ à partir des critères de détection et renvoie les potentiels candidats ayant le minimum de $(5, 3)$ et $(6, 4)$. Cette nouvelle fonction coût est placée après la fonction coût de minimisation de cycles. Nous obtenons ainsi un algorithme RandPEG non quasi-cyclique amélioré, noté RandPEGnoTS, qui minimise les cycles et les trois trapping-sets caractérisés. Pour la minimisation des TSEs $(6, 4)$ nous privilégions la minimisation des $(6, 4; 8^2, 12^1)$, ce choix est dû au fait que tous les TSEs $\mathcal{T}(a, b)$ avec $7 \leq a \leq 8$ et $b \leq 4$ sont des successeurs des $\mathcal{T}(5, 3; 8^3)$ ou $\mathcal{T}(6, 4; 8^2, 12^1)$ cf. figure 3.6. L'algorithme peut être utilisé avec l'un quelconque des modes suivants ou avec une combinason de certain d'entre eux :

- $min\,(nbCycles)$: Minimise les cycles
- $min\,(nbTS53)$: Minimise les trapping-sets $\left(5, 3; 8^3\right)$
- $min\,(nbTS64_1)$: Minimise les trapping-sets $\left(6, 4; 8^2, 12^1\right)$
- $min\,(nbTS64_2)$: Minimise les trapping-sets $\left(6, 4; 8^1, 10^2\right)$

Le RandPEGnoTS qui fait la combinaison de tous ces modes est défini comme suit :

Input : N, M, D_v, g_t
Output : G
for $n = 0$ **to** $N - 1$ **do**
  for $i = 0$ **to** $d_{v_n} - 1$ **do**
    if i=0 **then**
      $E^0_{v_n} \leftarrow branche\,(c_m, v_n)$, où $E^0_{v_n}$ est la première branche incidendente à v_n et c_m est le nœud de contrôle qui a le plus faible degré courant du sous graphe avec les branches existantes $E_{v_0} \cup E_{v_1} \cup E_{v_2} \cup \cdots E_{v_{n-1}}$.
    end
    else
      étendre, avec les branches existantes, le sous graphe depuis le nœud de variable v_n jusqu'à la profondeur $k = 5$ (ici $k = 5$, c'est pour gérer la minimisation des TS(6,4))
      if $\overline{\mathcal{M}}^k_{v_n} \neq \varnothing$ **then**
        Choix aléatoire d'un nœud de contrôle c_m dans $\overline{\mathcal{M}}^k_{v_n}$:
        $E^i_{v_n} \leftarrow branche\,(c_m, v_n)$ (pas de cycle de longueur g_t créé)
      end
      if $\overline{\mathcal{M}}^k_{v_n} = \varnothing$ **then**
        if $\mathcal{M}^k_{v_n} \neq \varnothing$ **then**
          Etape 1 : Elimine les nœuds de profondeur inférieure à $k < g_t/2 - 1$
          Etape 2 : Elimine tous les nœuds de contrôle qui créent plus de cycles que la valeur minimale des occurences $min\,(nbCycles_{c_m})$
          Etape 3 : Si $g_t = 8$ Elimine tous les nœuds de contrôle qui créent plus de $min\,(nbtrapping_{c_m})$
          Etape 4 : Si $g_t = 8$ Elimine tous les nœuds de contrôle qui créent plus de $min_m(nombrede(6, 4; 8^2, 12^1))$
          Etape 5 : Si $g_t = 8$ Elimine tous les nœuds de contrôle qui créent plus de $min_m(nombrede(6, 4; 8^1, 10^2))$
          Etape 6 : Mettre à jour $\mathcal{M}^k_{v_n}$
        end
        if $\mathcal{M}^k_{v_n} \neq \varnothing$ **then**
          Choix aléatoire d'un nœud de contrôle c_m dans $\mathcal{M}^k_{v_n}$:
          $E^i_{v_n} \leftarrow branche\,(c_m, v_n)$
        end
        else
          Echec de l'algorithme
        end
      end
    end
  end
end

Algorithme 4: Algorithme RandPEGnoTS non-QC

En guise d'exemple, les choix se font comme sur les figures 3.14 et 3.15 où il existe déjà un 8-cycle.

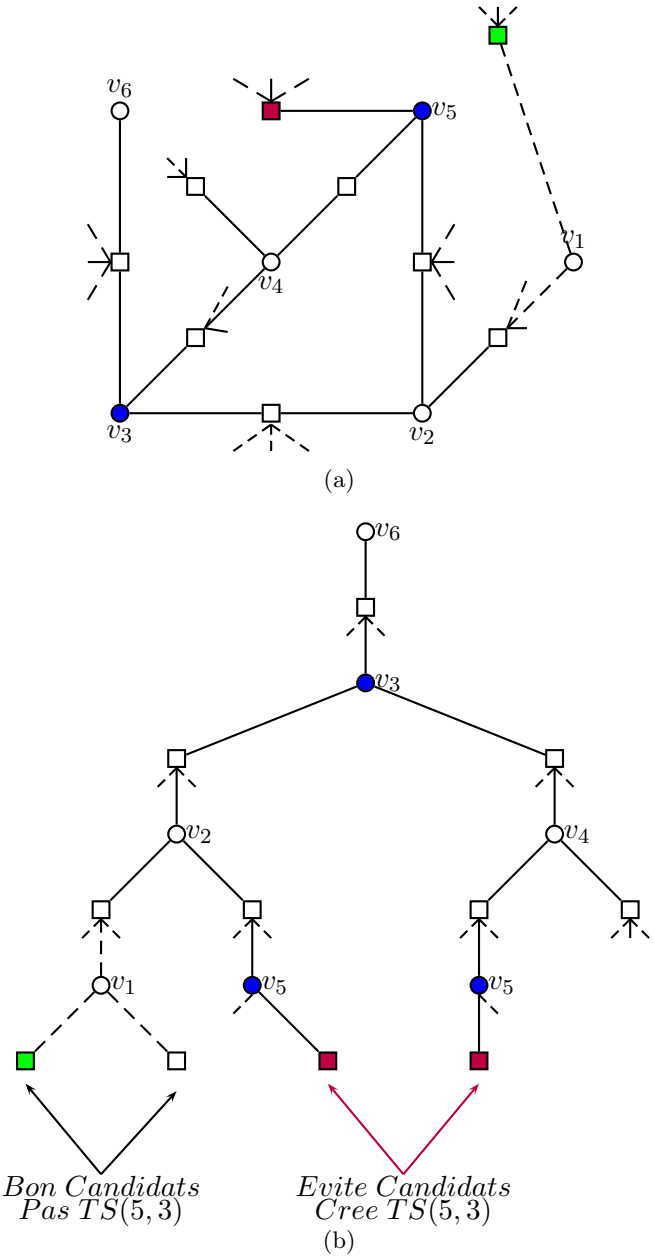


FIGURE 3.14 – Choix pour éviter un $(5,3;8^3)$

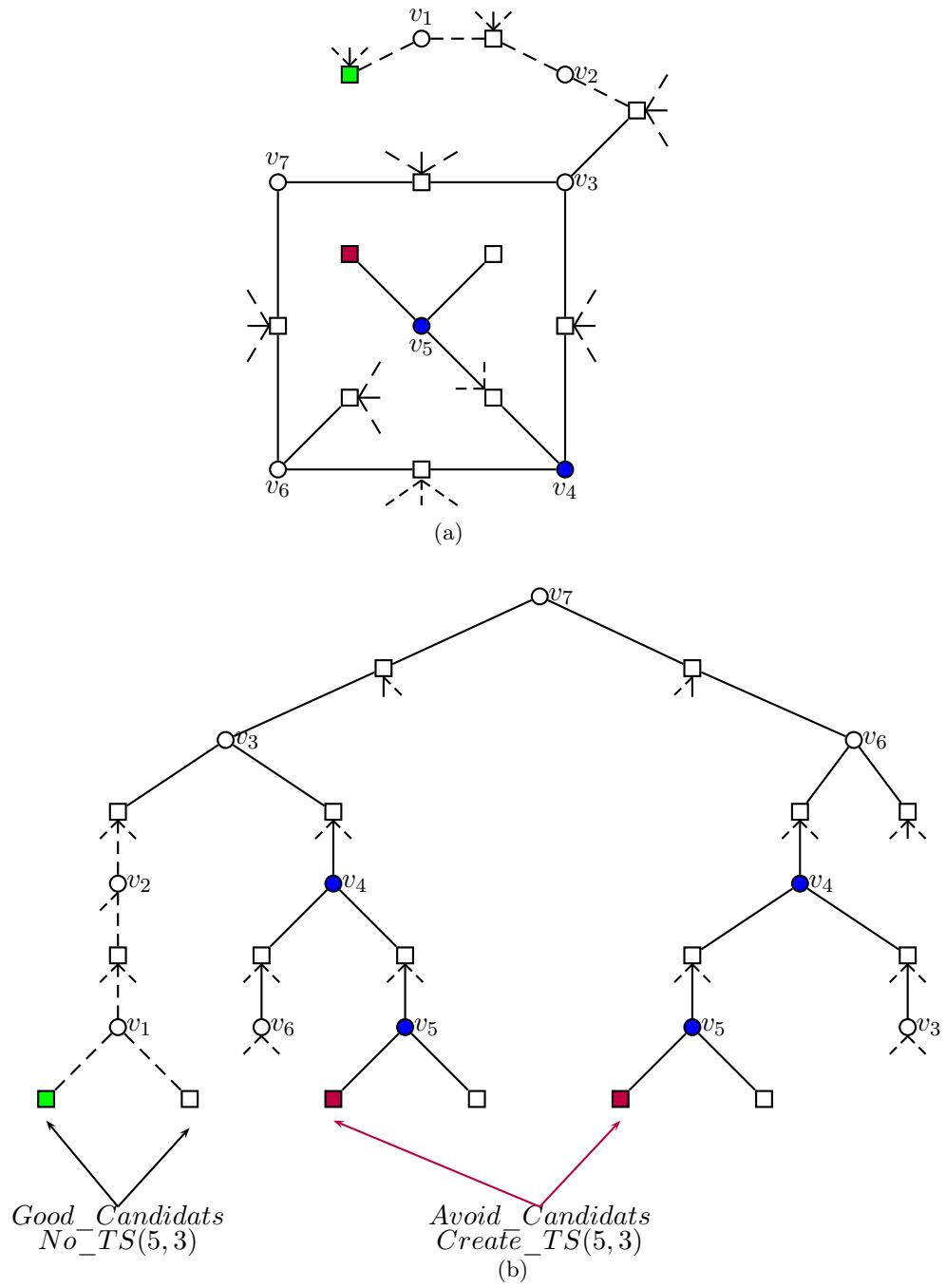


FIGURE 3.15 – Choix pour éviter un $(5, 3; 8^3)$

Sur les simulations effectuées pour les codes non-quasi cycliques, nous utilisons uniquement la combinaison des modes $min\,(nbCycles)$ et $min\,(nbTS53)$. Sur 10^5 tests du RandPEGnoTS pour un code LDPC régulier non quasi-cyclique $\mathcal{C}(155, 3, 5)$, nous avons 4813 succès. Le nombre de TS$(5, 3; 8^3)$ sur les matrices obtenues varie entre 0 et 15 sous forme gaussienne comme le montre la figure 3.16. Un

résultat interessant à noter que l'histogramme montre est qu'avec ces paramètres du code, il existe bien des code non quasi-cyclique sans trapping-set $(5, 3; 8^3)$ c'est à dire $min\,(nbTS53) = 0$.

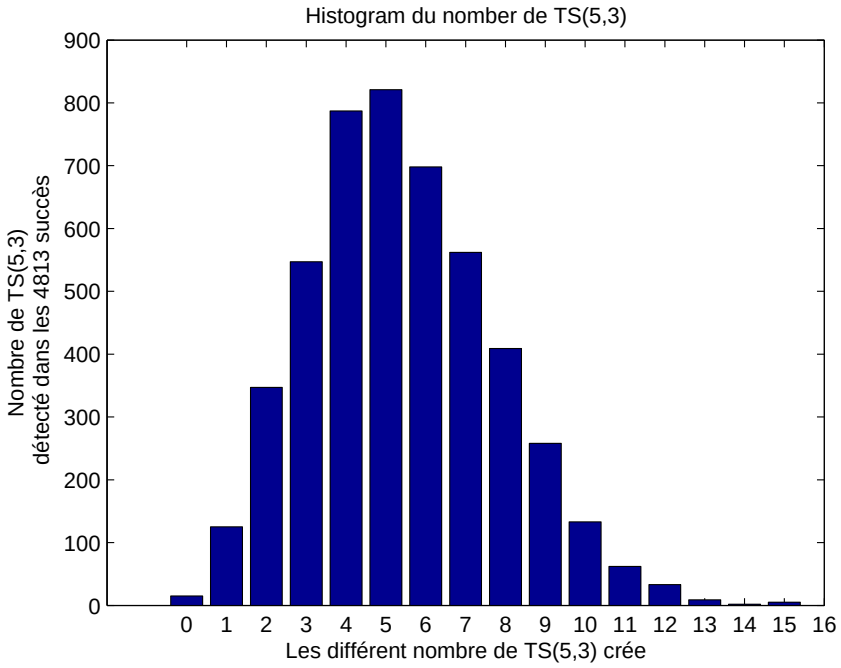


FIGURE 3.16 – Histogramme du Nombre de TS$(5, 3; 8^3)$ sur les 4813 matrices

La figure 3.17, montre que les trapping-sets sont créés lors de la création des branches des 30 derniers nœuds de variable et pour ces nœuds il existe peu de potentiels candidats pour les créations de la deuxième branche et de la troisième branche comme le montre la figure 3.18. La figure 3.18 montre que le nombre de potentiels candidats décroît par intervalle de N/d_c. Pour chaque variable v_n, où $n = l.N/d_c$ nous avons un maximum local des potentiels candidats.

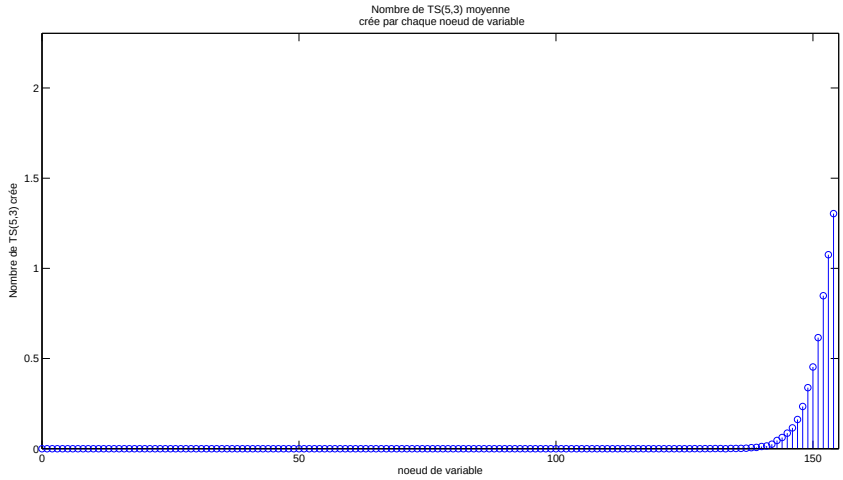


FIGURE 3.17 – Nombre de trapping-set $(5, 3; 8^3)$ créé par chaque nœud de variable

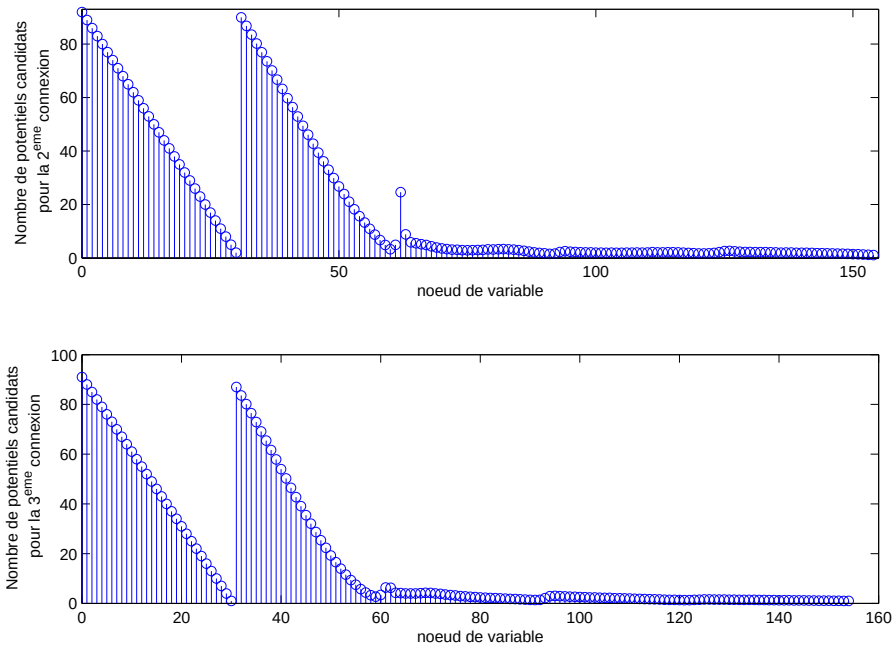


FIGURE 3.18 – Nombre de candidat pour la 2^{eme} et 3^{eme} branche de chaque nœud de variable

Avec les codes non quasi-cycliques le nombre de potentiels candidats est important pour les noeuds de variable v_n, où $n = l.N/d_c$ et qu'il existe des codes non quasi-cycliques sans des TSEs $(5, 3; 8^3)$. Ces résultats nous ont poussé à fixer comme contrainte l'élimination totale des trapping-sets $(5, 3; 8^3)$ avec une mini-

misation des TSEs $(6, 4)$ lors de la conception des codes QC-LDPC qui sont très utilisés dans les applications pratiques.

3.3.2 RandPEGnoTS pour la conception de codes QC-LDPC

L'algorithme RandPEGnoTS s'adapte facilement pour les codes QC-LDPC, il suffit avec les matrices de permutation circulantes de prendre un "saut de pas" de p lors de la conception c'est à dire étendre l'arbre pour les noeuds de variable $v_{l.p}$ seulement. Il faut aussi prendre en compte les cycles non-vus en intégrant la fonction coût qui élimine les 8-cycles et les 10-cycles. L'algorithme pour les codes QC-LDPC est :

Input : d_c, p, D_v, g_t
Output : G
for $n = 0$ **to** $d_c(p-1)$ **by gap** p **do**
  for $i = 0$ **to** $d_{v_n} - 1$ **do**
    if *i=0 or n=0* **then**
      $E^0_{v_n} \leftarrow branche\,(c_{i.p}, v_n)$, met la matrice identité sur la première ligne et sur la première colonne
    end
    else
      étendre, avec les branches existantes, le sous graphe depuis le nœud de variable v_n jusqu'à la profondeur $k = 5$ (ici $k = 5$, c'est pour gérer la minimisation des TS(6,4))
      if $\overline{\mathcal{M}}^k_{v_n} \neq \varnothing$ **then**
        Retire de $\overline{\mathcal{M}}^k_{v_n}$ tous les candidats qui ne satisfont pas les contraintes $\mathcal{C}o_8$ Mettre à jour $\overline{\mathcal{M}}^k_{v_n}$
      end
      if $\overline{\mathcal{M}}^k_{v_n} \neq \varnothing$ **then**
        Choix aléatoire d'un nœud de contrôle c_m dans $\overline{\mathcal{M}}^k_{v_n}$:
        $E^i_{v_n} \leftarrow branche\,(c_m, v_n)$ (pas de 8-cycle donc pas de (5,3) et de (6,4))
      end
      if $\overline{\mathcal{M}}^k_{v_n} = \varnothing$ **then**
        if $\mathcal{M}^k_{v_n} \neq \varnothing$ **then**
          Etape 1 : Elimine les nœuds de profondeur inférieure à 3
          Etape 2 : Elimine tous les nœuds de contrôle qui créent plus de cycles que la valeur minimale des cycles $min\,(nbCycles_{c_m})$
          Etape 3 : Retire de $\overline{\mathcal{M}}^k_{v_n}$ tous les candidats qui ne satisfont pas les contraintes $\mathcal{C}o_8$ Elimine tous les nœuds de contrôle qui donnent des 8-cycles non-vus
          Retire de $\overline{\mathcal{M}}^k_{v_n}$ tous les candidats qui ne satisfont pas les contraintes $\mathcal{C}o_{10}$ Elimine tous les nœuds de contrôle qui donnent des 10-cycles non-vus
          Etape 4 : Elimine tous les nœuds de contrôle qui créent des TS $(5, 3; 8^3)$
          Etape 5 : Elimine tous les nœuds de contrôle qui créent plus de $min_m(nombrede(6, 4; 8^2, 12^1))$
          Etape 6 : Elimine tous les nœuds de contrôle qui créent plus de $min_m(nombrede(6, 4; 8^1, 10^2))$
          Etape 7 : Mettre à jour $\mathcal{M}^k_{v_n}$
        end
        if $\mathcal{M}^k_{v_n} \neq \varnothing$ **then**
          Choix aléatoire d'un nœud de contrôle c_m dans $\mathcal{M}^k_{v_n}$:
          $E^i_{v_n} \leftarrow branche\,(c_m, v_n)$
        end
        else
          Echec de l'algorithme
        end
      end
    end
  end
end

Algorithme 5: Algorithme RandPEG QC-LDPC Sans TS$(5, 3; 8^3)$

Nous présentons sur le tableau 3.1 les valeurs minimales de la taille des circulantes en fonction du poids ligne d_c pour les codes QC-LDPC régulier $(3, d_c)$ sans TSE $(5, 3; 8^3)$, nous pouvons noter dans un premier temps que ces valeurs de p_{min} sont de loin inférieures aux bornes des codes de girth $g = 10$.

d_c	4	5	6	7	8	9	10
Borne conjecture	9	13	17	21	25	29	33
RandPEG	9	13	18	21	25	30	35
RandPEGnoTS$(5, 3; 8^3)$	12	18	23	31	40	52	65

TABLE 3.1 – p_{min} RandPEGnoTS pour les codes QC-LDPC régulier $(3, d_c)$ de girth $g_t = 8$

Après l'indentification des valeurs minimales de p en fonction de d_c pour les code QC-LDPC régulier $(3, d_c)$, nous donnons les statistiques de plusieurs constructions de code QC-LDPC régulier $(3, 5)$ dans le tableau 3.2.

	$p = 18$			$p = 31$				$p = 50$	
	PEG	Rand-PEG	No (5,3)	PEG	Rand-PEG	No (5,3)	Tanner	Rand-PEG	No (5,3)
$\#cycle - 6$	36	0	0	0	0	0	0	0	0
$\#cycle - 8$	684	774	720	682	620	527	465	575	325
$\#cycle - 10$	3402	3402	3582	3255	3348	3689	3720	2850	3350
$\#(5, 3; 8^3)$	0	144	0	62	31	0	155	50	0
$\#(6, 4; 8^2, 12)$	0	2286	1998	1271	930	434	0	550	0
$\#(6, 4; 8, 10^2)$	0	1530	1872	620	992	620	930	350	100

TABLE 3.2 – Comparaison des nombres de cycles et des nombres de trapping-sets pour plusieurs constructions de code QC-LDPC réguliers $(3, 5)$

Pour tous ces cas, les codes obtenus avec le nouveau RandPEG ne contien-nent pas de $(5, 3; 8^3)$ et présentent moins de TSEs $(6, 4)$ excepté pour $p = 18$ où nous avons plus de $(6, 4; 8^1, 10^2)$. Toujours pour le cas $p = 18$, le girth avec l'algorithme PEG obtenu est $g = 6$, mais il est démontré qu'avec des constructions PEG-like plus efficace, une girth $g = 8$ est atteinte pour ce rendement et cette longueur ce qui est le cas avec l'algorithme RandPEG. Dans le cas où $p = 31$, nous donnons aussi les statistiques du code de Tanner correspondant à la matrice 1.8 du chapitre 1 dont sa particularité est sa distance minimale $d_{min} = 20$. Pour $p = 50$, en plus d'éliminer les trapping-sets $(5, 3; 8^3)$, nous n'avons pas de trapping-sets $(6, 4; 8^2, 12^1)$ ce qui implique qu'il n'ya pas de TSE $(7, b)$ ni de TSE $(8, b)$ avec $b \leq 4$ (voir figure 3.6).

Les performances de ces codes du tableau 3.2 sont montrées sur les figures 3.19, 3.20. Ces simulations sont effectuées sur un canal BPSK-AWGN avec un décodeur BP utilisant 50 itérations au maximum. Les matrices construites avec le

RandPEGnoTS ont de meilleures performances. Noter que pour ces mots codes de longueur faible, nous n'avons pas atteint la région du plancher d'erreurs dans nos simulations (excepté pour le PEG, $p = 18$ code qui a un girth 6), mais l'amélioration de la pente est significative et montre que nous avons supprimé beaucoup de points fixes dominants du décodeur itératif.

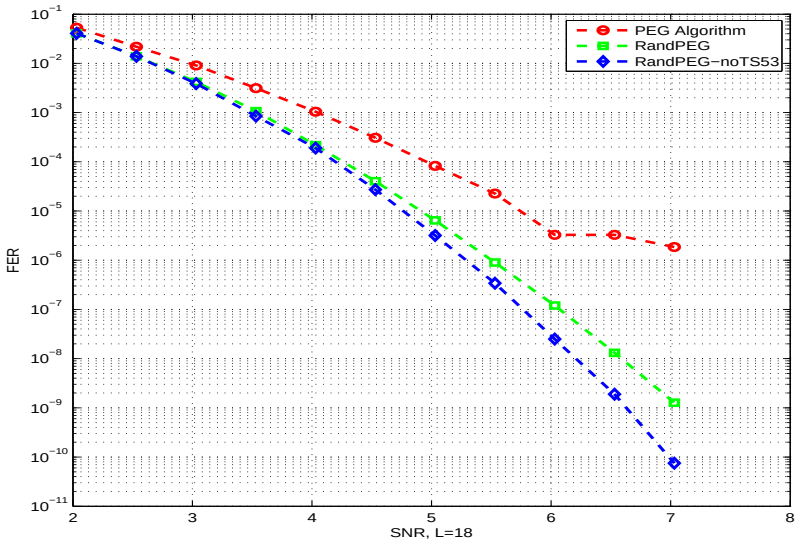


FIGURE 3.19 – Comparaison des performances des QC-LDPC avec $d_v = 3$, $d_c = 5$, $p = 18$

3.4 Un algorithme RandPEG minimisant les expansions minimales : RandPEGnoExp

Pour généraliser la détection prédictive des topologies nocives pour les décodeurs itératifs, nous utilisons les expansion-sets. Nous caractérisons les expansion-sets puis nous introduisons une approche pour la détection prédective d'une expansion-set élémentaire à partir de l'arbre des voisins.

3.4.1 Caractérisation des $E(a, b)$

Dans un arbre tous les nœuds qui figurent sur un chemin ont au moins deux voisins excepté les feuilles. Ainsi, pour la détection d'une expansion-set élémentaire $E(a, b)$ à partir d'un arbre nous avons besoin du nombre de nœuds de contrôle de degré deux dans l'expansion, de l'ensemble des nœuds de variable $\mathcal{A}(c_m)$ et de l'ensemble des nœuds de contrôle ascendants pour chaque nœud. Nous dénotons

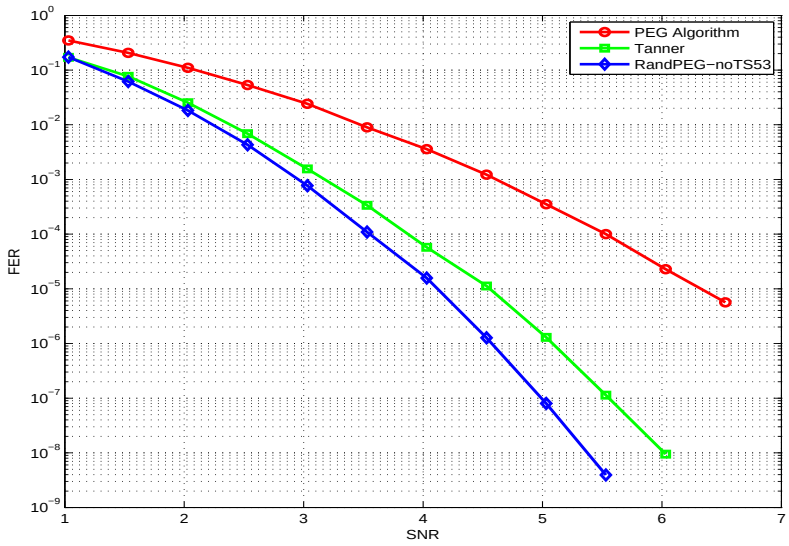


FIGURE 3.20 – Comparaison des performances des QC-LDPC avec $d_v = 3$, $d_c = 5$, $p = 31$

b_e le nombre de nœuds de contrôle de degré **deux** dans une expansion-set élémentaire $E(a, b)$, b_o le nombre de nœuds de contrôle de degré **un**, $\mathcal{A}_c(c_{m_i}^{l_i})$ l'ensemble des nœuds de contrôle ascendants du nœud de contrôle $c_{m_i}^{l_i}$ et nous utilisons la notation $E_e(a, b_e; \prod_{k\geq 2}(2k)^{g_{2k}})$, pour expliciter que nous prenons en considération que les nœuds de contrôle de degré deux sur les critères de détection d'expansion $E(a, b; \prod_{k\geq 2}(2k)^{g_{2k}})$.

Théorème 3.8 : *Soit un code $\mathcal{C}_{d_v,g_t}$, pour tout nœud de variable v_n une expansion-set de type $E_e(a, b_e; \prod_{k\geq 2}(2k)^{g_{2k}})$ est détectée dans T_{v_n} si et seulement si il existe $\beta = \lfloor \frac{-1+\sqrt{1+8\alpha}}{2} \rfloor$ copies de c_m qui ont les mêmes parents, avec $\alpha = \sum_{k\geq 2} g_{2k}$ tel que :*

- $\left| \bigcup_{i=1}^{\beta} \mathcal{A}\left(c_{m_i}^{l_i}\right) \right| = a$
- $\left| \bigcup_{i=1}^{\beta} \mathcal{A}_c\left(c_{m_i}^{l_i}\right) \right| = b_e$
- *et que l'ensemble maximum des $\frac{\beta(\beta+1)}{2}$ cycles formés avec la connexion (v_n, c_m) soit contenu dans l'ensemble des α cycles de l'expansion,*
  $\left(ie.\ \frac{\beta(\beta+1)}{2}\ cycles \subseteq \prod_{k\geq 2}(2k)^{g_{2k}}\right)$.

Preuve : *D'après le lemme 3.2, il faut $\beta = \lfloor \frac{-1+\sqrt{1+8\alpha}}{2} \rfloor$ copies d'un nœud de contrôle pour détecter une combinaison de α cycles. Si les copies n'ont pas les même parents, alors l'expansion n'est pas élémentaire.*

- *Si* $\left|\bigcup_{i=1}^{\beta}\mathcal{A}\left(c_{m_i}^{l_i}\right)\right| \neq a$, *alors nous n'avons pas une expansion-set avec* **a** *nœuds de variable.*
- *Si* $\left|\bigcup_{i=1}^{\beta}\mathcal{A}_c\left(c_{m_i}^{l_i}\right)\right| \neq b_e$, *alors n'avons pas une expansion-set avec* b_e *nœuds de contrôle de degré deux.*

Maintenant supposons que $\left|\bigcup_{i=1}^{\beta}\mathcal{A}\left(c_{m_i}^{l_i}\right)\right| = a$ *et* $\left|\bigcup_{i=1}^{\beta}\mathcal{A}_c\left(c_{m_i}^{l_i}\right)\right| = b_e$. *Si l'ensemble des cycles formés avec la connexion de* (v_n, c_m) *est inclus ou égale à l'ensemble des* α *cycles de l'expansion alors nous avons l'expansion ou un de ses prédécesseurs. Mais une expansion-set et son prédécesseur ne peuvent pas avoir simultanément la même valeur de* **a** *et de* b_e. *Finalement si* $\left|\bigcup_{i=1}^{\beta}\mathcal{A}\left(c_{m_i}^{l_i}\right)\right| = a$, $\left|\bigcup_{i=1}^{\beta}\mathcal{A}_c\left(c_{m_i}^{l_i}\right)\right| = b_e$ *et l'ensemble des cycles formés avec la connexion de* (v_n, c_m) *est inclus ou égale à l'ensemble des* α *cycles de l'expansion alors nous avons l'expansion* $E_e(a, b_e; \prod_{k\geq 2}(2k)^{g_{2k}})$.

Remarque 3.1 *Les longueurs des* $\frac{\beta(\beta-1)}{2}$ *cycles sont calculées en utilisant le théorème 3.4 et les différents cas sont toutes les* **différentes combinaisons** *de* β *niveaux* l_i *de la forme* $l_i = 2k-1$ *avec* $g_{2k} \neq 0$.

Nous utilisons ce théorème pour donner les critères de détection d'une topologie avec $d_v \neq 3$. Nous traitons l'exemple du $AS\,(4,4)$ représenté sur la figure 3.21 où $d_v = 4$. Le $AS\,(4,4)$ est composé de 4 cycles de longueur 6 et de 3 cycles de longueur 8 donc l'$AS\,(4,4)$ est l'équivalent d'une expansion $E\left(4, 10; 6^4, 8^3\right)$ notée aussi $E_e\left(4, 6; 6^4, 8^3\right)$ pour un code $\mathcal{C}_{4,6}$. L'expansion $E\left(4, 10; 6^4, 8^3\right)$ est un successeur direct de l'expansion $E\left(4, 11; 6^2, 8^1\right)$ qui est formé de trois cycle, cette expansion $E\left(4, 11; 6^2, 8^1\right)$ est un successeur direct de $E(3, 9; 6^1)$ qui correspond à un 6-cycle et aussi un successeur direct de $E(4, 12; 8^1)$ qui correspond à un 8-cycle. Pour les codes $\mathcal{C}_{4,6}$, l'expansion-set $E\left(4, 10; 6^4, 8^3\right)$ est parmi les expansions les plus nocives [DZA+10] et nous donnons sur le corollaire suivant les critères de sa détection prédictive.

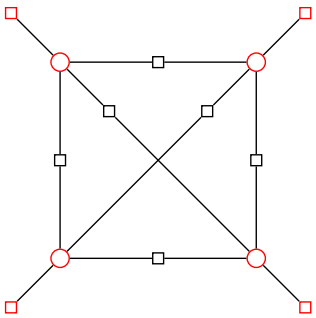


FIGURE 3.21 – Expansion $E\left(4, 10; 6^4, 8^3\right)$ équivalent au $AS(4,4)$

Corollaire 3.2 : *Soit un code $\mathcal{C}_{4,6}$, pour tout nœud de variable v_n une expansion $E(4,10;6^4,8^3)$ notée aussi $E_e(4,6;6^4,8^3)$ est détectée si et seulement s'il existe $\beta = 3$ copies de c_m tel que :*

- $\left|\bigcup_{i=1}^{3} \mathcal{A}\left(c_{m_i}^{l_i}\right)\right| = 4$
- $\left|\bigcup_{i=1}^{3} \mathcal{A}_c\left(c_{m_i}^{l_i}\right)\right| = 6$
- *et que l'ensemble maximum des $\frac{\beta(\beta+1)}{2} = 6$ cycles formés par la connexion (v_n, c_m) soit contenu dans 4 cycles de longueurs 6 et 3 cycles de longueurs 8*

Les niveaux des 3 copies sont un des deux cas :

- $1^{er}Cas$ **:** *il ya deux copies de niveau 5 et une copie de niveau 7 (exemple $l_1 = 5, l_2 = 5, l_3 = 7$)*
- $2^{eme}Cas$ **:** *il ya une copie de niveau 5 et deux copies de niveau 7 (exemple $l_1 = 5, l_2 = 2, l_3 = 7$)*

3.4.2 Algorithme RandPEG minimisant les expansions

De manière générale, l'algorithme pour la minimisation des expansions est identique à celui du RandPEGnoTS où il faut remplacer la fonction de coût minimisant les $(5,3)$ et $(6,4)$ par la fonction coût minimisant les expansions. La profondeur d'arrêt dans l'arbre dépend souvent de la taille du plus grand cycle présent dans la topologe. En faisant ce changement pour les $E\left(4,10;6^4,8^3\right)$, nous obtenons l'algorithme défini ci-dessous. Avec ce nouveau algorithme nous avons eu des codes QC-LDPC réguliers $(4,d_c)$ sans $E\left(4,10;6^4,8^3\right)$ avec des valeurs minimales faibles qui sont données sur le tableau 3.3. Ces résultats qui sont très proches des résultats mininales d'un code QC-LDPC réguliers $(4,d_c)$ de girth 6 sans gestion des AS(4,4) dont les valeurs étaient données dans le tableau 2.5 et reprises ici. En particulier, pour d_c pair les bornes obtenues avec l'algorithme RandPEG sans $E\left(4,10;6^4,8^3\right)$ sont optimales et sont identiques aux bornes minimales obtenues avec le RandPEG original.

d_c	5	6	7	8	9	10
Borne minimale	5	7	7	9	9	11
p_{min} RandPEG	5	7	7	10	10	11
p_{min} sans $E\left(4,10;6^4,8^3\right)$	7	7	9	10	11	11

TABLE 3.3 – Valeurs minimales de p pour les codes QC-LDPC réguliers $(4,d_c)$ de girth $g = 6$ avec et sans $E\left(4,10;6^4,8^3\right)$

Input : d_c, p, D_v, g_t
Output : G
for $n = 0$ **to** $d_c(p-1)$ **by gap** p **do**
  for $i = 0$ **to** $d_{v_n} - 1$ **do**
    if *i=0 or n=0* **then**
      $E^0_{v_n} \leftarrow branche\,(c_{i.p}, v_n)$, met la matrice identité sur la première ligne et sur la première colonne
    end
    else
      étendre, avec les branches existantes, le sous graphe depuis le nœud de variable v_n jusqu'à la profondeur k
      if $\overline{\mathcal{M}}^k_{v_n} \neq \varnothing$ **then**
        Retire de $\overline{\mathcal{M}}^k_{v_n}$ tous les candidats qui ne satisfont pas les contraintes $\mathcal{C}o_8$ Mettre à jour $\overline{\mathcal{M}}^k_{v_n}$
      end
      if $\overline{\mathcal{M}}^k_{v_n} \neq \varnothing$ **then**
        Choix aléatoire d'un nœud de contrôle c_m dans $\overline{\mathcal{M}}^k_{v_n}$:
        $E^i_{v_n} \leftarrow branche\,(c_m, v_n)$ (pas de 6-cycle donc pas de $E\left(4, 10; 6^4, 8^3\right)$)
      end
      if $\overline{\mathcal{M}}^k_{v_n} = \varnothing$ **then**
        if $\mathcal{M}^k_{v_n} \neq \varnothing$ **then**
          Etape 1 : Elimine les nœuds de profondeur inférieure à 3
          Etape 2 : Elimine tous les nœuds de contrôle qui créent plus de cycles que la valeur minimale des cycles $min\,(nbCycles_{c_m})$
          Etape 3 : Retire de $\overline{\mathcal{M}}^k_{v_n}$ tous les candidats qui ne satisfont pas les contraintes $\mathcal{C}o_8$ Elimine tous les nœuds de contrôle qui donnent des 8-cycles non-vus
          Retire de $\overline{\mathcal{M}}^k_{v_n}$ tous les candidats qui ne satisfont pas les contraintes $\mathcal{C}o_{10}$ Elimine tous les nœuds de contrôle qui donnent des 10-cycles non-vus
          Etape 4 : Elimine tous les nœuds de contrôle qui créent l'expansion non-désirée (ici c'est le $E\left(4, 10; 6^4, 8^3\right)$)
          Etape 7 : Mettre à jour $\mathcal{M}^k_{v_n}$
        end
        if $\mathcal{M}^k_{v_n} \neq \varnothing$ **then**
          Choix aléatoire d'un nœud de contrôle c_m dans $\mathcal{M}^k_{v_n}$:
          $E^i_{v_n} \leftarrow branche\,(c_m, v_n)$
        end
        else
          Echec de l'algorithme
        end
      end
    end
  end
end

Algorithme 6: Algorithme RandPEG QC-LDPC Sans Expansion

Dans le tableau 3.4 nous présentons les statistiques des cycles et des expansion-sets $E\left(4,10;6^4,8^3\right)$ pour des codes QC-LDPC réguliers $(4,d_c)$ avec d_c pair. Les codes quasi-cycliques de même valeur pour d_c ont sensiblement les même nombres de 6-cycles tandis que les codes obtenus avec l'algorithme RandPEG original donnent moins de 8-cycles que le code sans $E\left(4,10;6^4,8^3\right)$. Par contre, les codes sans $E\left(4,10;6^4,8^3\right)$ ont les meilleures performances comme le montre la figure 3.22. Ces simulations sont effectuées sur un canal AWGN, et le récepteur utilise l'algorithme BP avec 50 itérations au maximum.

	$p=7,\ d_c=6$		$p=10,\ d_c=8$		$p=11,\ d_c=10$	
	Rand-PEG	RandPEG sans $E\left(4,10;6^4,8^3\right)$	Rand-PEG	RandPEG sans $E\left(4,10;6^4,8^3\right)$	Rand-PEG	RandPEG sans $E\left(4,10;6^4,8^3\right)$
$\#cycle-6$	672	672	1700	1720	3520	3520
$\#cycle-8$	7056	7308	26895	26990	73502	73920
$\#E\left(4,10;6^4,8^3\right)$	126	0	140	0	209	0

TABLE 3.4 – Comparaison des codes de girth 6 avec des $E\left(4,10;6^4,8^3\right)$ et des codes de girth 6 sans $E\left(4,10;6^4,8^3\right)$

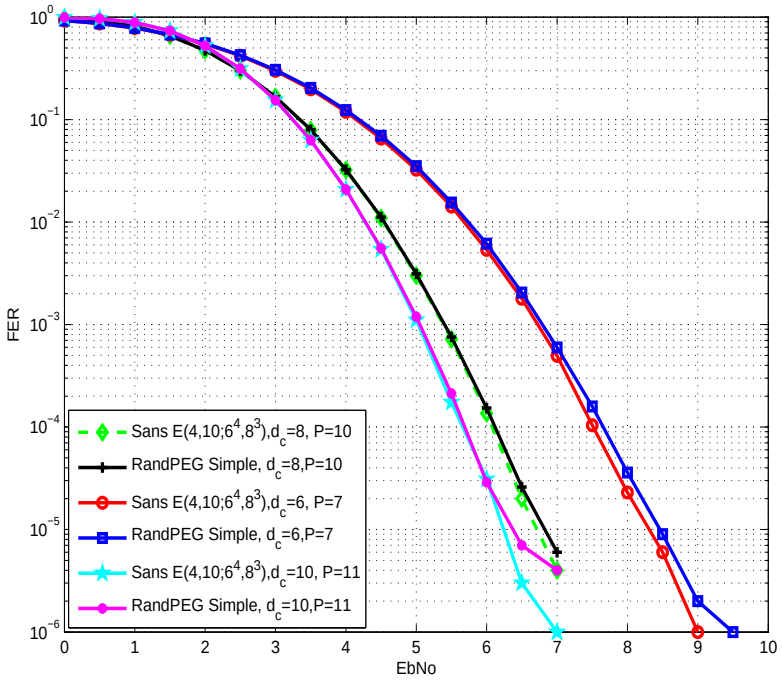


FIGURE 3.22 – Comparaison des performances de code QC-LDPC régulier (4,6) de girth 6

3.5 Conclusion

Dans ce chapitre nous avons rappelé les topologies nocives pour les décodeurs itératifs. Nous nous sommes plus concentrés sur les TSE (5,3) et (6,4) qui sont les trapping-sets dominants pour un code $\mathcal{C}_{3,8}$. Nous avons proposé des critéres de détections prédictives de ces trapping-sets puis nous avons utilisé ces critères sur l'algorithme RandPEG afin de contruire des codes sans TSE (5,3) avec une minimisation des TSEs (6,4). Nous avons terminé par une généralisation de la détection prédictive des topologies nocives avec le concept d'expansion. Pour les codes $\mathcal{C}_{4,6}$, nous avons contruit des codes QC-LDPC réguliers $(4, d_c)$ sans $E\left(4, 10; 6^4, 8^3\right)$ dont les résultats sont sensiblement identiques aux bornes des codes QC-LDPC réguliers $(4, d_c)$ construits avec l'algorithme PEG et qui comportent des absorbing sets AS(4,4).

Conclusion Générale

Dans cette thèse, nous nous intéressons à la conception optimale/optimisée des codes LDPC binaires réguliers à faible plancher d'erreurs. Nous avons proposé trois contributions principales sur la conception des codes LDPC. La première contribution est une amélioration de la borne minimale pour les codes QC-LDPC régulier $(3, d_c)$. La deuxième contribution est un algorithme qui permet la conception optimale de code QC-LDPC avec de large girth et de taille faible. La troixième contribution est un algorithme qui permet la conception optimisée des codes LDPC réguliers (QC, protographs) en minimisant les trapping-sets ou les expansion-sets dominantes. Cette minimisation s'effectue par une détection prédictive des topologies dominantes définies. Par des comparaisons de taille minimimale et des courbes de performances sur le taux d'erreurs bloc, nous avons prouvé l'efficacité de nos méthodes proposées.

Le chapitre 1 constitue l'état de l'art sur la conception des codes LDPC. A partir de la desciption des méthodes existantes, nous avons remarqué que toutes ces méthodes ont pour objectif principal l'augmentation du girth et/ou la réduction de la complexité d'encodage et de décodage. Nous avons constaté aussi que toutes les constructions structurées peuvent se convertir en code QC-LDPC qui sont favorables dans l'implémentation matérielle et que les constructions aléatoires donnent de faible plancher d'erreurs. Ces remarques ont motivé notre choix de réaliser des méthodes de construction aléatoire utilisant des matrices de permutation circulante.

Dans le chapitre 2, nous avons effectué un rappel des bornes minimales pour les matrices de permutation circulante des codes QC-LDPC réguliers (d_v, d_c). A l'issue d'une analyse faite, nous avons donné par conjecture une nouvelle borne minimale pour les matrices de permutation circulante d'un code QC-LDPC $\mathcal{C}_{3,8}$. La contribution principale dans ce chapitre est l'introduction de la notion de *cycle non-vu* dans l'algorithme PEG et ses dérivées pour les codes QC-LDPC. Les critères de détection de ces cycles non-vus sont donnés dans les corollaires 2.6 à 2.17. Ces critères sont appliqués dans l'algorithme RandPEG modifié que nous avons

proposé. Cet algorithme permet de construire des codes de girth inférieur ou égale à 12 avec les meilleures bornes minimales. De plus, pour les valeurs faibles de d_c l'algorithme permet d'atteindre les bornes minimales théoriques définies pour les codes QC-LDPC réguliers $(3, d_c)$ de girth inférieur où égale à 10.

Dans le chapitre 3, nous avons proposé dans un premier temps des critères de détection prédictive des trapping-sets (5,3) et (6,4) pour un code QC-LDPC régulier $(3, d_c)$ de girth 8. Ces critères sont appliqués sur l'algorithme RandPEG pour la conception de code LDPC sans les TSEs (5,3) avec une minimisation des TSEs (6,4). Dans la phase de minimisation des TSEs (6,4), nous avons privilégié la minimisation des TSEs $(6, 4; 8^2, 12^1)$ sur les TSEs $(6, 4; 8^1, 10^2)$. Ce choix est motivé par leurs différents successeurs dans la hiérarchie des TSEs d'un code $\mathcal{C}_{3,8}$. Dans un deuxième temps, les critères de détection prédictive sont généralisés pour toute topologie élémentaire dont les cycles qui la forment sont connus. L'ensemble de ces critères sont les principales contributions pour ce chapitre. Des simulations avec différents rendements ont été effectuées pour prouver l'avantage des codes LDPC qui ne présentent pas certaines topologies nocives.

Les contributions présentées dans ce manuscrit ont fait l'objet de quelques publications dans des conférences internationales listées en annexe A.

Perspectives

A partir de ce travail, quelques points méritent d'être développés dans un futur proche. Parmi ces points nous avons :

- La preuve de la conjecture 2.1 proposée dans le chapitre 2. Ce qui permettrait de considérer la conjecture comme un théorème.
- La généralisation des bornes atteignables pour les codes QC-LDPC (d_v, d_c) de girth $g \geq 8$. En effet, les bornes minimales atteignables ne sont pas définies pour les codes QC-LDPC $(3, d_c)$ de girth 12 et les codes QC-LDPC (d_v, d_c) avec $d_v > 3$.
- L'impact des trapping-sets/expansion-sets sur les nouveaux décodeurs tel que le FAID (Finite alphabet iterative decoders) et le decimation-enhanced FAID.
- L'application des algorithmes proposés pour la constructions des codes QC-LDPC irréguliers. En effet, dans tout le travail effectué seuls les codes réguliers ont été considéré.

ANNEXES

Annexe A

Publications

CI$_1$ **M. DIOUF**, D. DECLERCQ, S. OUYA, and B. VASIC, "*A PEG-like LDPC code design avoiding short Trapping Sets,*" IEEE International Symposium on Information Theory (ISIT2015), Hong Kong, Chine, 14-19 June, 2015.

CI$_2$ S.K. PLANJERY, D. DECLERCQ, **M. DIOUF**, B.VASIC, "*On the Guaranteed Error Correction of Decimation-Enhanced Iterative Decoders,*" in Proc. IEEE International Symposium on Turbo Codes and Iterative Information Processing (ISTC8), Bremen, Germany, 18-22 Aug., 2014, pp 57-61.

CN$_1$ **M. DIOUF**, D. DECLERCQ, S. OUYA, and B. VASIC, "*Conception de Code LDPC avec l'Algorithme PEG en Evitant les Trapping Sets Court,*" GRETSI, Lyon, France, Septembre 2015.

CN$_2$ **M. DIOUF**, D. DECLERCQ, and S. OUYA, "*Amélioration de l'Algorithme PEG par Minimisation des Trapping-Set TS(5,3) sur les Codes LDPC,*" Colloque National de la Recherche en Informatique et ses Applications (CNRIA), Ziguinchor, Senegal, Avril 2013.

Annexe B

Cardinal des ensembles

Dans cette partie les ensembles sont ceux définis dans la preuve du théorème 2.9. Nous donnons les cardinaux en fonction du girth avec la condition $0 \neq l \neq L-1$.

- $S_0 = \{0\} \Rightarrow |S_0| = 1$, $S_1' = \{p_{1,L-1}\} \Rightarrow \left|S_1'\right| = 1$, $S_2' = \{p_{2,L-1}\} \Rightarrow \left|S_2'\right| = 1$
- $S_1 = \{p_{1,l}\}$.
  $p_{1,l} \neq p_{1,l'}$, $\forall l \neq l^{'}$ si et seulement si $p_{1,l} - p_{0,l} + p_{0,l'} - p_{1,l'} \neq 0$ ie $g \geq 6$ donc si $g \geq 6$ alors $|S_1| = L-2$.
- $S_2 = \{p_{2,l}\}$.
  $p_{2,l} \neq p_{2,l'}$, $\forall l \neq l^{'}$ si et seulement si $p_{2,l} - p_{0,l} + p_{0,l'} - p_{2,l'} \neq 0$ ie $g \geq 6$ donc si $g \geq 6$ alors $|S_2| = L-2$.
- $S_3 = \{p_{2,L-1} + p_{1,l} - p_{2,l}\}$.
  $p_{2,L-1} + p_{1,l} - p_{2,l} \neq p_{2,L-1} + p_{1,l'} - p_{2,l'}$, $\forall l \neq l^{'}$ si et seulement si $p_{1,l} - p_{2,l} + p_{2,l'} - p_{1,l'} \neq 0$ ie $g \geq 6$ donc si $g \geq 6$ alors $|S_3| = L-2$.
- $S_4 = \{p_{2,L-1} - p_{2,l}\}$.
  $p_{2,L-1} - p_{2,l} \neq p_{2,L-1} - p_{2,l'}$ si et seulement si $p_{2,l'} - p_{0,l'} + p_{0,l} - p_{2,l} \neq 0$ ie $g \geq 6$ donc si $g \geq 6$ alors $|S_4| = (L-2)$.
- $S_5 = \{p_{1,L-1} - p_{1,l}\}$.
  $p_{1,L-1} - p_{1,l} \neq p_{1,L-1} - p_{1,l'}$ si et seulement si $p_{1,l} - p_{0,l} + p_{0,l'} - p_{1,l'} \neq 0$ ie $g \geq 6$ donc si $g \geq 6$ alors $|S_5| = (L-2)$.

- $S_6 = \{p_{1,l} - p_{2,l}\}$.
  $p_{1,l} - p_{2,l} \neq p_{1,l'} - p_{2,l'}$ si et seulement si $p_{1,l} - p_{2,l} + p_{2,l'} - p_{1,l'} \neq 0$ ie $g \geq 6$ donc si $g \geq 6$ alors $|S_6| = (L-2)$.
- $S_7 = \{p_{2,l} - p_{1,l}\}$.
  $p_{2,l} - p_{1,l} \neq p_{2,l'} - p_{1,l'}$ si et seulement si $p_{2,l} - p_{1,l} + p_{1,l'} - p_{2,l'} \neq 0$ ie $g \geq 6$ donc si $g \geq 6$ alors $|S_7| = (L-2)$.
- $S_8 = \left\{p_{2,L-1} + p_{1,l_{k_2}} - p_{2,l_{k_1}}\right\}$ avec $l_{k_1} \neq l_{k_2}$.
  $p_{2,L-1} + p_{1,l_{k_2}} - p_{2,l_{k_1}} \neq p_{2,L-1} + p_{1,l'_{k_2}} - p_{2,l'_{k_1}}$ si et seulement si $p_{1,l_{k_2}} - p_{0,l_{k_2}} + p_{0,l_{k_1}} - p_{2,l_{k_1}} + p_{2,l'_{k_1}} - p_{0,l'_{k_1}} + p_{0,l'_{k_2}} - p_{1,l'_{k_2}} \neq 0$ ie $g \geq 10$ donc si $g \geq 10$ alors $|S_8| = (L-2)(L-3)$. Mais pour l_{k_1} fixé, $p_{2,L-1} + p_{1,l_{k_2}} - p_{2,l_{k_1}} \neq p_{2,L-1} + p_{1,l'_{k_2}} - p_{2,l_{k_1}}$ si et seulement si $p_{1,l_{k_2}} - p_{0,l_{k_2}} + p_{0,l'_{k_2}} - p_{1,l'_{k_2}} \neq 0$ ie $g \geq 6$ donc si $g \geq 6$ alors $|S_8| = (L-3)$, si l_{k_1} est fixé.
- $S_9 = \left\{p_{1,l_{k_1}} - p_{2,l_{k_1}} + p_{2,l_{k_2}}\right\}$ avec $l_{k_1} \neq l_{k_2}$.
  $p_{1,l_{k_1}} - p_{2,l_{k_1}} + p_{2,l_{k_2}} \neq p_{1,l'_{k_1}} - p_{2,l'_{k_1}} + p_{2,l'_{k_2}}$ si et seulement si $p_{1,l_{k_1}} - p_{2,l_{k_1}} + p_{2,l_{k_2}} - p_{0,l_{k_2}} + p_{0,l'_{k_2}} - p_{2,l'_{k_2}} + p_{2,l'_{k_1}} - p_{1,l'_{k_1}} \neq 0$ ie $g \geq 10$ donc si $g \geq 10$ alors $|S_9| = (L-2)(L-3)$. Mais pour l_{k_2} fixé, $p_{1,l_{k_1}} - p_{2,l_{k_1}} + p_{2,l_{k_2}} \neq p_{1,l'_{k_1}} - p_{2,l'_{k_1}} + p_{2,l_{k_2}}$ si et seulement si $p_{1,l_{k_1}} - p_{2,l_{k_1}} + p_{2,l'_{k_1}} - p_{1,l'_{k_1}} \neq 0$ ie $g \geq 6$ donc si $g \geq 6$ alors $|S_9| = (L-3)$, si l_{k_2} est fixé.
- $S_{10} = \left\{p_{2,l_{k_1}} - p_{1,l_{k_1}} + p_{1,l_{k_2}}\right\}$ avec $l_{k_1} \neq l_{k_2}$.
  $p_{2,l_{k_1}} - p_{1,l_{k_1}} + p_{1,l_{k_2}} \neq p_{2,l'_{k_1}} - p_{1,l'_{k_1}} + p_{1,l'_{k_2}}$ si et seulement si $p_{2,l_{k_1}} - p_{1,l_{k_1}} + p_{1,l_{k_2}} - p_{0,l_{k_2}} + p_{0,l'_{k_2}} - p_{1,l'_{k_2}} + p_{1,l'_{k_1}} - p_{2,l'_{k_1}} \neq 0$ ie $g \geq 10$ donc si $g \geq 10$ alors $|S_{10}| = (L-2)(L-3)$. Mais pour l_{k_2} fixé, $p_{2,l_{k_1}} - p_{1,l_{k_1}} + p_{1,l_{k_2}} \neq p_{2,l'_{k_1}} - p_{1,l'_{k_1}} + p_{1,l_{k_2}}$ si et seulement si $p_{2,l_{k_1}} - p_{1,l_{k_1}} + p_{1,l'_{k_1}} - p_{2,l'_{k_1}} \neq 0$ ie $g \geq 6$ donc si $g \geq 6$ alors $|S_{10}| = (L-3)$, si l_{k_2} est fixé.

Annexe C

Relation entre tous les ensembles S_n

Dans cette partie nous donnons les conditions pour que deux ensembles soient disjoints. Ces conditions sont exprimés en fonction du girth et nous nous sommes limités aux ensembles S_i avec $i \leq 4$.

- $S_1 \bigcap S_0 = \varnothing$ si et seulement si $p_{1,l} \neq 0 \Rightarrow g \geq 6$. Donc pour $g = 8,\ S_1 \bigcap S_0 = \varnothing$
- Pour S_2 nous avons :
  1. $S_2 \bigcap S_0 = \varnothing$ si et seulement si $p_{2,l} \neq 0 \Rightarrow g \geq 6$. Donc pour $g = 8,\ S_2 \bigcap S_0 = \varnothing$
  2. $S_2 \bigcap S_1 = \varnothing$ si et seulement si $p_{2,l} \neq p_{1,l'} \Rightarrow g \geq 8$. Donc pour $g = 8,\ S_2 \bigcap S_1 = \varnothing$.

  Par suite, $\left| \bigcup_{i=0}^{2} S_i \bigcup S_1^{'} \bigcup S_2^{'} \right| = 2\,(L-1) + 1$.
- Pour S_3 nous avons :
  1. $S_3 \bigcap S_0 = \varnothing$ si et seulement si $p_{2,L-1} + p_{1,l} - p_{2,l} \neq 0$ ie. $p_{2,L-1} - p_{0,L-1} + p_{0,0} - p_{1,0} + p_{1,l} - p_{2,l} \neq 0 \Rightarrow g \geq 8$. Donc pour $g = 8,\ S_3 \bigcap S_0 = \varnothing$.
  2. $S_3 \bigcap S_1 = \varnothing$ si et seulement si $p_{2,L-1} + p_{1,l} - p_{2,l} \neq p_{1,l'}$ ie. $p_{2,L-1} - p_{0,L-1} + p_{0,l'} - p_{1,l'} + p_{1,l} - p_{2,l} \neq 0 \Rightarrow g \geq 8$. Donc pour $g = 8,\ S_3 \bigcap S_1 = \varnothing$.
  3. $S_3 \bigcap S_1^{'} = \varnothing$ si et seulement si $p_{2,L-1} + p_{1,l} - p_{2,l} \neq p_{1,L-1}$ ie. $p_{2,L-1} - p_{1,L-1} + p_{1,l} - p_{2,l} \neq 0 \Rightarrow g \geq 6$. Donc pour $g = 8,\ S_3 \bigcap S_1^{'} = \varnothing$.
  4. $S_3 \bigcap S_2 = \varnothing$ si et seulement si $p_{2,L-1} + p_{1,l} - p_{2,l} \neq p_{2,l'}$ ie. $p_{2,L-1} - p_{0,L-1} + p_{0,l'} - p_{2,l'} + p_{2,0} - p_{1,0} + p_{1,l} - p_{2,l} \neq 0 \Rightarrow g \geq 10$. Donc pour $g = 10,\ S_3 \bigcap S_2 = \varnothing$. Mais $S_3 \bigcap S_2^{'} = \varnothing$ si $g \geq 6$.

De plus si nous supposons le cas où $S_3 \bigcap S_2 = \varnothing$ pour $g \geq 8$, alors nous obtenons $\left| \bigcup_{i=0}^{3} \bigcup S_1' \bigcup S_2' \right| = 3\,(L-1)$.

– Pour S_4 et une approche similaire aux précédentes, nous avons :

1. $S_4 \bigcap S_0 = \varnothing$ si et seulement si $p_{2,L-1} - p_{2,l} \neq 0 \Rightarrow g \geq 6$. Donc pour $g = 8,\ S_4 \bigcap S_0 = \varnothing$.
2. $S_4 \bigcap S_1 = \varnothing$ si et seulement si $p_{2,L-1} - p_{2,l} \neq p_{1,l'}$ ie. $p_{2,L-1} - p_{0,L-1} + p_{0,l'} - p_{1,l'} + p_{1,0} - p_{0,0} + p_{0,l} - p_{2,l} \neq 0 \Rightarrow g \geq 10$. Donc pour $g = 10,\ S_4 \bigcap S_1 = \varnothing$. Mais pour $g = 8,\ S_4 \bigcap S_1' = \varnothing$.
3. $S_4 \bigcap S_2 = \varnothing$ si et seulement si $p_{2,L-1} - p_{2,l} \neq p_{2,l'}$ ie. $p_{2,L-1} - p_{0,L-1} + p_{0,l'} - p_{2,l'} + p_{2,0} - p_{0,0} + p_{0,l} - p_{2,l} \neq 0 \Rightarrow g \geq 10$. Donc pour $g = 10,\ S_4 \bigcap S_2 = \varnothing$. Mais pour $g \geq 6,\ S_4 \bigcap S_2' = \varnothing$.
4. $S_4 \bigcap S_3 = \varnothing$ si et seulement si $p_{2,L-1} - p_{2,l} \neq p_{2,L-1} + p_{1,l'} - p_{2,l'}$ ie. $p_{1,l'} - p_{2,l'} + p_{2,l} - p_{0,l} + p_{0,0} - p_{1,0} \neq 0 \Rightarrow g \geq 8$. Donc pour $g = 8,\ S_4 \bigcap S_3 = \varnothing$.

Annexe D

Quelques Exemples de Matrices de Bases

Dans cette section, nous donnons quelques matrices de bases qui ont de meilleures tailles minimales p. Chaque coefficient $H_{B_{j,l}}$ de H_B représente la valeur du décalage de la matrice de permutation circulante $I_{p_{j,l}}$.

D.1 Matrices de tailles Minimales

D.1.1 Girth $g = 8$ et $d_v = 3$

– $p = 40$ & $d_c = 11$

$$H_B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 9 & 15 & 2 & 22 & 7 & 34 & 5 & 18 & 28 & 38 \\ 0 & 30 & 4 & 13 & 6 & 1 & 8 & 10 & 35 & 37 & 25 \end{bmatrix}$$

– $p = 46$ & $d_c = 12$

$$H_B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 12 & 8 & 30 & 31 & 1 & 21 & 19 & 18 & 11 & 4 & 33 \\ 0 & 2 & 39 & 17 & 28 & 13 & 25 & 38 & 29 & 14 & 26 & 10 \end{bmatrix}$$

D.1.2 Girth g=10 et $d_v = 3$

- $p = 37$ & $d_c = 4$

$$H_B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 29 & 13 & 30 \\ 0 & 6 & 18 & 33 \end{bmatrix}$$

- $p = 61$ & $d_c = 5$

$$H_B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 42 & 53 & 28 \\ 0 & 60 & 20 & 2 & 54 \end{bmatrix}$$

- $p = 91$ & $d_c = 6$

$$H_B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 69 & 34 & 23 & 16 \\ 0 & 51 & 81 & 32 & 27 & 90 \end{bmatrix}$$

D.2 Matrices sans Trapping-sets (5,3) avec $d_v = 3$, $d_c = 5$ et g=8

- $p = 18$

$$H_B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 12 & 13 & 2 & 17 \\ 0 & 15 & 9 & 6 & 7 \end{bmatrix}$$

- $p = 31$

$$H_B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 27 & 24 & 18 & 19 \\ 0 & 25 & 8 & 20 & 10 \end{bmatrix}$$

- $p = 50$ ***et sans*** $TS(6, 4; 8^2, 12^1)$

$$H_B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 24 & 41 & 28 & 43 \\ 0 & 3 & 30 & 42 & 48 \end{bmatrix}$$

D.3 Matrices sans trapping-sets (5,3) et (6,4) avec $g = 8$ et $d_v = 3$

- $p = 37$ ***et*** $d_c = 4$

$$H_B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 11 & 30 & 9 \\ 0 & 33 & 32 & 19 \end{bmatrix}$$

- $p = 61$ ***et*** $d_c = 5$

$$H_B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 40 & 31 & 27 & 60 \\ 0 & 23 & 28 & 8 & 6 \end{bmatrix}$$

- $p = 91$ ***et*** $d_c = 6$

$$H_B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 21 & 24 & 20 & 9 & 51 \\ 0 & 50 & 10 & 12 & 45 & 28 \end{bmatrix}$$

D.4 Matrice sans $E(4, 10; 6^4, 8^3)$ de taille Minimale avec $d_v = 4$ et $g = 6$

- $p = 7$ ***et*** $d_c = 6$

$$H_B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & 3 & 5 & 2 & 4 \\ 0 & 2 & 1 & 4 & 3 & 6 \\ 0 & 5 & 6 & 3 & 4 & 1 \end{bmatrix}$$

- $p = 10$ ***et*** $d_c = 8$

$$H_B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 8 & 2 & 4 & 7 & 5 & 6 & 3 \\ 0 & 3 & 6 & 1 & 8 & 7 & 4 & 9 \\ 0 & 4 & 5 & 8 & 6 & 2 & 7 & 1 \end{bmatrix}$$

– $p = 11$ ***et*** $d_c = 10$

$$H_B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 8 & 4 & 9 & 10 & 6 & 1 & 5 & 3 & 2 \\ 0 & 6 & 3 & 4 & 2 & 10 & 9 & 1 & 5 & 7 \\ 0 & 2 & 1 & 5 & 8 & 7 & 3 & 4 & 9 & 6 \end{bmatrix}$$

Bibliographie

[802] IEEE Std 802.16e. Air interface for fixed and mobile broadband wireless access systems.

[ABAA11] R. Asvadi, A.H. Banihashemi, and M. Ahmadian-Attari. Lowering the error floor of ldpc codes using cyclic liftings. *Information Theory, IEEE Transactions on*, 57(4) :2213–2224, April 2011.

[ABAA12] R. Asvadi, A.H. Banihashemi, and M. Ahmadian-Attari. Design of finite-length irregular protograph codes with low error floors over the binary-input awgn channel using cyclic liftings. *Communications, IEEE Transactions on*, 60(4) :902–907, April 2012.

[AHK+04] B. Ammar, B. Honary, Yu Kou, Jun Xu, and Shu Lin. Construction of low-density parity-check codes based on balanced incomplete block designs. *Information Theory, IEEE Transactions on*, 50(6) :1257–1269, June 2004.

[ASDDR10] S. Abu-Surra, D. Declercq, D. Divsalar, and W.E. Ryan. Trapping set enumerators for specific ldpc codes. In *Information Theory and Applications Workshop (ITA), 2010*, pages 1–5, Jan 2010.

[BCH08] N. Bonello, Sheng Chen, and L. Hanzo. Construction of regular quasi-cyclic protograph ldpc codes based on vandermonde matrices. *Vehicular Technology, IEEE Transactions on*, 57(4) :2583–2588, July 2008.

[Ber10] C. Berrou. Codes and turbo codes. In *Collection IRIS, Springer-Verlag Paris*, volume 1, May 2010.

[BGT93] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error-correcting coding and decoding : Turbo-codes. 1. In *Communications, 1993. ICC '93 Geneva. Technical Program, Conference Record, IEEE International Conference on*, volume 2, pages 1064–1070 vol.2, May 1993.

[BTCN11] S. Bandi, V. Tralli, A. Conti, and M. Nonato. On girth conditioning for low-density parity-check codes. *Communications, IEEE Transactions on*, 59(2) :357–362, February 2011.

[CBW10] Chao Chen, Baoming Bai, and Xinmei Wang. Construction of nonbinary quasi-cyclic ldpc cycle codes based on singer perfect difference set. *Communications Letters, IEEE*, 14(2) :181–183, February 2010.

[CC07] Hua Chen and Zhigang Cao. A modified peg algorithm for construction of ldpc codes with strictly concentrated check-node degree distributions. In *Wireless Communications and Networking Conference, 2007.WCNC 2007. IEEE*, pages 564–568, March 2007.

[CFRU01] Sae-Young Chung, Jr. Forney, G.D., T.J. Richardson, and R. Urbanke. On the design of low-density parity-check codes within 0.0045 db of the shannon limit. *Communications Letters, IEEE*, 5(2) :58–60, Feb 2001.

[CKV08] S.K. Chilappagari, A.R. Krishnan, and B. Vasic. Ldpc codes which can correct three errors under iterative decoding. In *Information Theory Workshop, 2008. ITW '08. IEEE*, pages 406–410, May 2008.

[CMR01] J. Campello, D.S. Modha, and S. Rajagopalan. Designing ldpc codes using bit-filling. In *Communications, 2001. ICC 2001. IEEE International Conference on*, volume 1, pages 55–59 vol.1, Jun 2001.

[CNVM10a] S.K. Chilappagari, D.V. Nguyen, B. Vasic, and M.W. Marcellin. Error correction capability of column-weight-three ldpc codes under the gallager a algorithm ;part ii. *Information Theory, IEEE Transactions on*, 56(6) :2626–2639, June 2010.

[CNVM10b] S.K. Chilappagari, D.V. Nguyen, B. Vasic, and M.W. Marcellin. On trapping sets and guaranteed error correction capability of ldpc codes and gldpc codes. *Information Theory, IEEE Transactions on*, 56(4) :1600–1611, April 2010.

[CSV06] S.K. Chilappagari, S. Sankaranarayanan, and B. Vasic. Error floors of ldpc codes on the binary symmetric channel. In *Communications, 2006. ICC '06. IEEE International Conference on*, volume 3, pages 1089–1094, June 2006.

[CV09] S.K. Chilappagari and B. Vasic. Error-correction capability of column-weight-three ldpc codes. *Information Theory, IEEE Transactions on*, 55(5) :2055–2061, May 2009.

[CVSC09] S.K. Chilappagari, B. Vasic, M. Stepanov, and M. Cherktov. Analysis of error floor of ldpc codes under lp decoding over the bsc. In *in Proc. IEEE International Symposium on Information Theory (ISIT'09)*, pages 379–383, July 2009.

[CZD+14] Fang Cai, Xinmiao Zhang, D. Declercq, S.K. Planjery, and B. VasicÌĄ. Finite alphabet iterative decoders for ldpc codes : Optimization, architecture and analysis. *Circuits and Systems I : Regular Papers, IEEE Transactions on*, 61(5) :1366–1375, May 2014.

[DB10] M.K. Dehkordi and A.H. Banihashemi. An efficient algorithm for finding dominant trapping sets of ldpc codes. In *Turbo Codes and Iterative Information Processing (ISTC), 2010 6th International Symposium on*, pages 444–448, Sept 2010.

[DDM98] H. Divsalar D., Jin and R. McEliece. Coding theorems for turbo-like codes. In *Proceeding of the 36th Allerton conference on communication, control and computing*, 1998.

[DDPV11] L. Danjean, D. Declercq, S.K. Planjery, and B. Vasic. On the selection of finite alphabet iterative decoders for ldpc codes on the bsc. In *Information Theory Workshop (ITW), 2011 IEEE*, pages 345–349, Oct 2011.

[DF07] D. Declercq and M. Fossorier. Decoding algorithms for nonbinary ldpc codes over gf (q). *Communications, IEEE Transactions on*, 55(4) :633–643, April 2007.

[DF08] D. Declercq and M. Fossorier. Improved impulse method to evaluate the low weight profile of sparse binary linear codes. In *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*, pages 1963–1967, July 2008.

[DLZ+09] L. Dolecek, P. Lee, Zhengya Zhang, V. Anantharam, B. Nikolic, and M. Wainwright. Predicting error floors of structured ldpc codes : deterministic bounds and

estimates. *Selected Areas in Communications, IEEE Journal on*, 27(6) :908–917, August 2009.

[Dol10] L. Dolecek. On absorbing sets of structured sparse graph codes. In *Information Theory and Applications Workshop (ITA), 2010*, pages 1–5, Jan 2010.

[DPT+02] Changyan Di, D. Proietti, I.E. Telatar, T.J. Richardson, and R.L. Urbanke. Finite-length analysis of low-density parity-check codes on the binary erasure channel. *Information Theory, IEEE Transactions on*, 48(6) :1570–1579, Jun 2002.

[DS] Draft DVB-S2. Standard.

[DTLAG12] Qiuju Diao, Ying Yu Tai, Shu Lin, and K. Abdel-Ghaffar. Trapping set structure of finite geometry ldpc codes. In *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, pages 3088–3092, July 2012.

[DVPL13] D. Declercq, B. Vasic, S.K. Planjery, and Erbao Li. Finite alphabet iterative decoders-part ii : Towards guaranteed error correction of ldpc codes via iterative decoder diversity. *Communications, IEEE Transactions on*, 61(10) :4046–4057, October 2013.

[DWZ10] L. Dolecek, Jiadong Wang, and Zhengya Zhang. Towards improved ldpc code designs using absorbing set spectrum properties. In *Turbo Codes and Iterative Information Processing (ISTC), 2010 6th International Symposium on*, pages 477–481, Sept 2010.

[DXAGL03] I. Djurdjevic, Jun Xu, K. Abdel-Ghaffar, and Shu Lin. A class of low-density parity-check codes constructed based on reed-solomon codes with two information symbols. *Communications Letters, IEEE*, 7(7) :317–319, July 2003.

[DZA+07] L. Dolecek, Zhengya Zhang, V. Anantharam, M. Wainwright, and B. Nikolic. Analysis of absorbing sets for array-based ldpc codes. In *Communications, 2007. ICC '07. IEEE International Conference on*, pages 6261–6268, June 2007.

[DZA+10] L. Dolecek, Zhengya Zhang, V. Anantharam, M.J. Wainwright, and B. Nikolic. Analysis of absorbing sets and fully absorbing sets of array-based ldpc codes. *Information Theory, IEEE Transactions on*, 56(1) :181–201, Jan 2010.

[DZLbL14] Le Dong, Ziyu Zhao, Jing Lei, and Er bao Li. Lower error floor of ldpc codes based on trapping sets elimination. In *Wireless Communications and Signal Processing (WCSP), 2014 Sixth International Conference on*, pages 1–5, Oct 2014.

[EG10] M. Esmaeili and M. Gholami. Structured quasi-cyclic {LDPC} codes with girth 18 and column-weight. *{AEU} - International Journal of Electronics and Communications*, 64(3) :202 – 217, 2010.

[Fan00] J. L. Fan. Array codes as low-density parity-check codes. *Proc. 2nd International Symposium on Turbo Codes and their applications*, pages 543–546, Sep 2000.

[Fos04] M.P.C. Fossorier. Quasicyclic low-density parity-check codes from circulant permutation matrices. *Information Theory, IEEE Transactions on*, 50(8) :1788–1793, Aug 2004.

[Gal62] R.G. Gallager. Low-density parity-check codes. *Information Theory, IRE Transactions on*, 8(1) :21–28, January 1962.

[GH12] A. Gruner and M. Huber. New combinatorial construction techniques for low-density parity-check codes and systematic repeat-accumulate codes. *Communications, IEEE Transactions on*, 60(9) :2387–2395, September 2012.

[GOS04] M. Greferath, M.E. O'Sullivan, and R. Smarandache. Construction of good ldpc codes using dilation matrices. In *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*, pages 237–, June 2004.

[HC06] T.R. Halford and K.M. Chugg. An algorithm for counting short cycles in bipartite graphs. *Information Theory, IEEE Transactions on*, 52(1) :287–292, Jan 2006.

[HdL12] C.T. Healy and R.C. de Lamare. Decoder-optimised progressive edge growth algorithms for the design of ldpc codes with low error floors. *Communications Letters, IEEE*, 16(6) :889–892, June 2012.

[HDLAG11] Qin Huang, Qiuju Diao, Shu Lin, and K. Abdel-Ghaffar. Trapping sets of structured ldpc codes. In *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, pages 1086–1090, July 2011.

[HEA01] Xiao-Yu Hu, E. Eleftheriou, and D.-M. Arnold. Progressive edge-growth tanner graphs. In *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, volume 2, pages 995–1001 vol.2, 2001.

[HEA02] Xiao-Yu Hu, E. Eleftheriou, and D.-M. Arnold. Irregular progressive edge-growth (peg) tanner graphs. In *Information Theory, 2002. Proceedings. 2002 IEEE International Symposium on*, pages 480–, 2002.

[HEA05] Xiao-Yu Hu, E. Eleftheriou, and D.M. Arnold. Regular and irregular progressive edge-growth tanner graphs. *Information Theory, IEEE Transactions on*, 51(1) :386–398, Jan 2005.

[HHY12] Jen-Fa Huang, Chun-Ming Huang, and Chao-Chin Yang. Construction of one-coincidence sequence quasi-cyclic ldpc codes of large girth. *Information Theory, IEEE Transactions on*, 58(3) :1825–1836, March 2012.

[HLZZ10] Jie Huang, Lei Liu, Wuyang Zhou, and Shengli Zhou. Large-girth nonbinary qc-ldpc codes of various lengths. *Communications, IEEE Transactions on*, 58(12) :3436–3447, December 2010.

[HY12] Yejun He and Jie Yang. Construction of qc-ldpc codes with girth larger than eight based on gpu. In *Wireless Communications Signal Processing (WCSP), 2012 International Conference on*, pages 1–6, Oct 2012.

[ICV08] M. Ivkovic, S.K. Chilappagari, and B. Vasic. Eliminating trapping sets in low-density parity-check codes by using tanner graph covers. *Information Theory, IEEE Transactions on*, 54(8) :3763–3768, Aug 2008.

[JKM00] H. Jin, A. Khandekar, and Robert J. McEliece. Irregular repeat accumulate codes, 2000.

[JL09] Xueqin Jiang and Moon Ho Lee. Large girth non-binary ldpc codes based on finite fields and euclidean geometries. *Signal Processing Letters, IEEE*, 16(6) :521–524, June 2009.

[JM00] Hui Jin and R.J. McEliece. General coding theorems for turbo-like codes. In *Information Theory, 2000. Proceedings. IEEE International Symposium on*, pages 120–, 2000.

[JW01] S.J. Johnson and Steven R. Weller. Regular low-density parity-check codes from combinatorial designs. In *Information Theory Workshop, 2001. Proceedings. 2001 IEEE*, pages 90–92, 2001.

[JW02] S.J. Johnson and Steven R. Weller. Codes for iterative decoding from partial geometries. In *Information Theory, 2002. Proceedings. 2002 IEEE International Symposium on*, pages 310–, 2002.

[JW05] S.J. Johnson and Steven R. Weller. Constructions for irregular repeat-accumulate codes. In *Information Theory, 2005. ISIT 2005. Proceedings. International Symposium on*, pages 179–183, Sept 2005.

[KAB12] S. Khazraie, R. Asvadi, and A.H. Banihashemi. A peg construction of finite-length ldpc codes with low error floor. *Communications Letters, IEEE*, 16(8) :1288–1291, August 2012.

[KB11] M. Karimi and A.H. Banihashemi. An efficient algorithm for finding dominant trapping sets of irregular ldpc codes. In *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, pages 1091–1095, July 2011.

[KB12] M. Karimi and A.H. Banihashemi. Efficient algorithm for finding dominant trapping sets of ldpc codes. *Information Theory, IEEE Transactions on*, 58(11) :6942–6958, Nov 2012.

[KB13] M. Karimi and A.H. Banihashemi. On the girth of quasi-cyclic protograph ldpc codes. *Information Theory, IEEE Transactions on*, 59(7) :4542–4552, July 2013.

[KB14] M. Karimi and A.H. Banihashemi. On characterization of elementary trapping sets of variable-regular ldpc codes. *Information Theory, IEEE Transactions on*, 60(9) :5188–5203, Sept 2014.

[KCY13] Kyung-Joong Kim, Jin-Ho Chung, and Kyeongcheol Yang. Bounds on the size of parity-check matrices for quasi-cyclic low-density parity-check codes. *Information Theory, IEEE Transactions on*, 59(11) :7288–7298, Nov 2013.

[Kel08] C.A. Kelley. On codes designed via algebraic lifts of graphs. In *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, pages 1254–1261, Sept 2008.

[KFC05] Jingyu Kang, Pingyi Fan, and Zhigang Cao. Flexible construction of irregular partitioned permutation ldpc codes with low, error floors. *Communications Letters, IEEE*, 9(6) :534–536, Jun 2005.

[KFL01] F.R. Kschischang, B.J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *Information Theory, IEEE Transactions on*, 47(2) :498–519, Feb 2001.

[KHP11] Saejoon Kim, Jun Heo, and Hyuncheol Park. Improved stopping set elimination by parity-check matrix extension of ldpc codes. *Communications Letters, IEEE*, 15(5) :557–559, May 2011.

[KKKS07] Sung-Ha Kim, Joon-Sung Kim, Dae-Son Kim, and Hong-Yeop Song. Ldpc code construction with low error floor based on the ipeg algorithm. *Communications Letters, IEEE*, 11(7) :607–609, July 2007.

[KLF01] Yu Kou, Shu Lin, and M.P.C. Fossorier. Low-density parity-check codes based on finite geometries : a rediscovery and new results. *Information Theory, IEEE Transactions on*, 47(7) :2711–2736, Nov 2001.

[KNCS06] Sunghwan Kim, Jong-Seon No, Habong Chung, and Dong-Joon Shin. On the girth of tanner (3,5) quasi-cyclic ldpc codes. *Information Theory, IEEE Transactions on*, 52(4) :1739–1744, April 2006.

[KNCS07a] Sunghwan Kim, Jong-Seon No, Habong Chung, and Dong-Joon Shin. Cycle analysis and construction of protographs for qc ldpc codes with girth larger than 12. In *Information Theory, 2007. ISIT 2007. IEEE International Symposium on*, pages 2256–2260, June 2007.

[KNCS07b] Sunghwan Kim, Jong-Seon No, Habong Chung, and Dong-Joon Shin. Quasi-cyclic low-density parity-check codes with girth larger than 12. *Information Theory, IEEE Transactions on*, 53(8) :2885–2891, Aug 2007.

[KPP+04] J.L. Kim, U.N. Peled, I. Perepelitsa, V. Pless, and S. Friedland. Explicit construction of families of ldpc codes with no 4-cycles. *Information Theory, IEEE Transactions on*, 50(10) :2378–2388, Oct 2004.

[KS07] K.M. Krishnan and P. Shankar. Computing the stopping distance of a tanner graph is np-hard. *Information Theory, IEEE Transactions on*, 53(6) :2278–2280, June 2007.

[KV98] R. Kotter and A. Vardy. Factor graphs : constructions, classification, and bounds. In *Information Theory, 1998. Proceedings. 1998 IEEE International Symposium on*, pages 14–, Aug 1998.

[KW08] C.A. Kelley and J.L. Walker. Ldpc codes from voltage graphs. In *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*, pages 792–796, July 2008.

[KW10] Gyu Bum Kyung and Chih-Chun Wang. Exhaustive search for small fully absorbing sets and the corresponding low error-floor decoder. In *Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on*, pages 739–743, June 2010.

[LASMS98] M.G. Luby, M. Amin Shokrolloahi, M. Mizenmacher, and D.A. Spielman. Improved low-density parity-check codes using irregular graphs and belief propagation. In *Information Theory, 1998. Proceedings. 1998 IEEE International Symposium on*, pages 117–, Aug 1998.

[LCLC08] Yi-Kai Lin, Chin-Lung Chen, Yen-Chin Liao, and Hsie-Chia Chang. Structured ldpc codes with low error floor based on peg tanner graphs. In *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, pages 1846–1849, May 2008.

[LFK08] Keke Liu, Zesong Fei, and Jingming Kuang. Novel algebraic constructions of non-binary structured ldpc codes over finite fields. In *Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th*, pages 1–5, Sept 2008.

[LHMH09] S. Laendner, T. Hehn, O. Milenkovic, and J.B. Huber. The trapping redundancy of linear block codes. *Information Theory, IEEE Transactions on*, 55(1) :53–63, Jan 2009.

[LK04] Zongwang Li and B.V.K.V. Kumar. A class of good quasi-cyclic low-density parity check codes based on progressive edge growth graph. In *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Eighth Asilomar Conference on*, volume 2, pages 1990–1994 Vol.2, Nov 2004.

[LM05] S. Landner and O. Milenkovic. Algorithmic and combinatorial analysis of trapping sets in structured ldpc codes. In *Wireless Networks, Communications and Mobile Computing, 2005 International Conference on*, volume 1, pages 630–635 vol.1, 2005.

[LMH10] S. Laendner, O. Milenkovic, and J.B. Huber. Characterization of small trapping sets in ldpc codes from steiner triple systems. In *Turbo Codes and Iterative Information Processing (ISTC), 2010 6th International Symposium on*, pages 93–97, Sept 2010.

[LMN04] Jin Lu, J.M.F. Moura, and U. Niesen. Grouping-and-shifting designs for structured ldpc codes with large girth. In *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*, pages 236–, June 2004.

[LMSS98] M. Luby, M. Mitzenmacher, A. Shokrollah, and D. Spielman. Analysis of low density codes and improved designs using irregular graphs. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 249–258, 1998.

[LMSS01] M.G. Luby, M. Mitzenmacher, M.A. Shokrollahi, and D.A. Spielman. Improved low-density parity check codes using irregular graphs. *Information Theory, IEEE Transactions on*, 47(2) :585–598, Feb 2001.

[LSCZ10] M. Lentmaier, A. Sridharan, Jr. Costello, D.J., and K.Sh. Zigangirov. Iterative decoding threshold analysis for ldpc convolutional codes. *Information Theory, IEEE Transactions on*, 56(10) :5274–5289, Oct 2010.

[LSL+06] Gianluigi Liva, Shumei Song, Lan Lan, Yifei Zhang, Shu Lin, and William E. Ryan. Design of ldpc codes : A survey and new results, 2006.

[Mac] D. MacKay. Online database of low-density parity-check codes,[online] available : http ://www.inference.phy.cam.ac.uk/mackay/codes/data.html.

[Mac99] D.J.C. MacKay. Good error-correcting codes based on very sparse matrices. *Information Theory, IEEE Transactions on*, 45(2) :399–431, Mar 1999.

[Mar82] G. A. Margulis. Explicit constructions of graphs without short cycles and low density codes. *Combinatorica*, 02 :71–78, Mar 1982.

[Mar88] G. A. Margulis. Explicit group-theoretical constructions of combinatorial schemes and their application to the design of expanders and concentrators. *Probl. Peredachi Inf.*, 24 :51–60, Mar 1988.

[MB01] Yongyi Mao and A.H. Banihashemi. A heuristic search for good low-density parity-check codes at short block lengths. In *Communications, 2001. ICC 2001. IEEE International Conference on*, volume 1, pages 41–44 vol.1, Jun 2001.

[MD99] D. J. C. MacKay and M. Davey. Evaluation of gallager code for short block length and high rate applications. *Proc. IMA Workshop on Codes, systems and Graphical Models*, 1999.

[MKL06] O. Milenkovic, N. Kashyap, and D. Leyba. Shortened array codes of large girth. *Information Theory, IEEE Transactions on*, 52(8) :3707–3722, Aug 2006.

[MN96] D.J.C. MacKay and R.M. Neal. Near shannon limit performance of low density parity check codes. *Electronics Letters*, 32(18) :1645–, Aug 1996.

[MSC11] D.G.M. Mitchell, R. Smarandache, and D.J. Costello. Quasi-cyclic ldpc codes based on pre-lifted protographs. In *Information Theory Workshop (ITW), 2011 IEEE*, pages 350–354, Oct 2011.

[MSW07] O. Milenkovic, E. Soljanin, and P. Whiting. Asymptotic spectra of trapping sets in regular and irregular ldpc code ensembles. *Information Theory, IEEE Transactions on*, 53(1) :39–55, Jan 2007.

[MWD99] D.J.C. MacKay, S.T. Wilson, and M.C. Davey. Comparison of constructions of irregular gallager codes. *Communications, IEEE Transactions on*, 47(10) :1449–1454, Oct 1999.

[MY05] Seho Myung and Kyeongcheol Yang. A combining method of quasi-cyclic ldpc codes by the chinese remainder theorem. *Communications Letters, IEEE*, 9(9) :823–825, Sep 2005.

[MYK05] Seho Myung, Kyeongcheol Yang, and Jaeyoel Kim. Quasi-cyclic ldpc codes for fast encoding. *Information Theory, IEEE Transactions on*, 51(8) :2894–2901, Aug 2005.

[MYK06] Seho Myung, Kyeongcheol Yang, and Youngkyun Kim. Lifting methods for quasi-cyclic ldpc codes. *Communications Letters, IEEE*, 10(6) :489–491, June 2006.

[NCMV10] Dung Viet Nguyen, Shashi Kiran Chilappagari, Michael W. Marcellin, and Bane V. Vasic. LDPC codes from latin squares free of small trapping sets. *CoRR*, abs/1008.4177, 2010.

[NCMV12] D.V. Nguyen, S.K. Chilappagari, M.W. Marcellin, and B. Vasic. On the construction of structured ldpc codes free of small trapping sets. *Information Theory, IEEE Transactions on*, 58(4) :2280–2302, April 2012.

[NVMC10] D.V. Nguyen, B. Vasic, M. Marcellin, and S.K. Chilappagari. Structured ldpc codes from permutation matrices free of small trapping sets. In *Information Theory Workshop (ITW), 2010 IEEE*, pages 1–5, Aug 2010.

[Oka03] T. Okamura. Designing ldpc codes using cyclic shifts. In *Information Theory, 2003. Proceedings. IEEE International Symposium on*, pages 151–, June 2003.

[O'S06] M.E. O'Sullivan. Algebraic construction of sparse matrices with large girth. *Information Theory, IEEE Transactions on*, 52(2) :718–727, Feb 2006.

[OUVZ02] A. Orlitsky, R. Urbanke, K. Viswanathan, and J. Zhang. Stopping sets and the girth of tanner graphs. In *Information Theory, 2002. Proceedings. 2002 IEEE International Symposium on*, pages 2–, 2002.

[PDCV10] Shiva Kumar Planjery, David Declercq, Shashi Kiran Chilappagari, and Bane Vasic. Multilevel decoders surpassing belief propagation on the binary symmetric channel. *CoRR*, abs/1001.3421, 2010.

[PDDV11] S.K. Planjery, D. Declercq, L. Danjean, and B. Vasic. Finite alphabet iterative decoders for ldpc codes surpassing floating-point iterative decoders. *Electronics Letters*, 47(16) :919–921, Aug 2011.

[PDDV13] S.K. Planjery, D. Declercq, L. Danjean, and B. Vasic. Finite alphabet iterative decoders-part i : Decoding beyond belief propagation on the binary symmetric channel. *Communications, IEEE Transactions on*, 61(10) :4033–4045, October 2013.

[PDDV14] S.K. Planjery, D. Declercq, M. Diouf, and B. Vasic. On the guaranteed error-correction of decimation-enhanced iterative decoders. In *Turbo Codes and Iterative Information Processing (ISTC), 2014 8th International Symposium on*, pages 57–61, Aug 2014.

[PFD08] C. Poulliat, M. Fossorier, and D. Declercq. Design of regular (2,d/sub c/)-ldpc codes over gf(q) using their binary images. *Communications, IEEE Transactions on*, 56(10) :1626–1635, October 2008.

[PHNS10] Hosung Park, Seokbeom Hong, Jong-Seon No, and Dong-Joon Shin. Protograph design for qc ldpc codes with large girth. In *Information and Communication*

Technology Convergence (ICTC), 2010 International Conference on, pages 175–176, Nov 2010.

[PHNS13] Hosung Park, Seokbeom Hong, Jong-Seon No, and Dong-Joon Shin. Design of multiple-edge protographs for qc ldpc codes avoiding short inevitable cycles. *Information Theory, IEEE Transactions on*, 59(7) :4598–4614, July 2013.

[PP11] E. Psota and L.C. Perez. Iterative construction of regular ldpc codes from independent tree-based minimum distance bounds. *Communications Letters, IEEE*, 15(3) :334–336, March 2011.

[RDV14] V. Ravanmehr, D. Declercq, and B. Vasic. Check-hybrid gldpc codes : Systematic elimination of trapping sets by super checks. In *Information Theory (ISIT), 2014 IEEE International Symposium on*, pages 701–705, June 2014.

[RDW15] S. V. S. Ranganathan, Dariush Divsalar, and Richard D. Wesel. On the girth of (3, L) quasi-cyclic LDPC codes based on complete protographs. *CoRR*, abs/1504.04975, 2015.

[RH06] G. Richter and A. Hof. On a construction method of irregular ldpc codes without small stopping sets. In *Communications, 2006. ICC '06. IEEE International Conference on*, volume 3, pages 1119–1124, June 2006.

[Ric03] T.J. Richardson. Error floors of ldpc codes. *Proc. 41st Annual Allerton Conf. on Communications, Control and Computing*, 2003.

[Ric06] Gerd Richter. Finding small stopping sets in the tanner graphs of ldpc codes. In *Turbo Codes Related Topics ; 6th International ITG-Conference on Source and Channel Coding (TURBOCODING), 2006 4th International Symposium on*, pages 1–5, April 2006.

[RSU00] T. Richardson, A. Shokrollahi, and R. Urbanke. Design of provably good low-density parity check codes. In *Information Theory, 2000. Proceedings. IEEE International Symposium on*, pages 199–, 2000.

[RSU01] T.J. Richardson, M.A. Shokrollahi, and R.L. Urbanke. Design of capacity-approaching irregular low-density parity-check codes. *Information Theory, IEEE Transactions on*, 47(2) :619–637, Feb 2001.

[RU01] T.J. Richardson and R.L. Urbanke. The capacity of low-density parity-check codes under message-passing decoding. *Information Theory, IEEE Transactions on*, 47(2) :599–618, Feb 2001.

[RU08] T. Richardson and R. Urbanke. Modern coding theory. In *Cambridge University Press*, 2008.

[RW04] A. Ramamoorthy and R. Wesel. Construction of short block length irregular low-density parity-check codes. In *Communications, 2004 IEEE International Conference on*, volume 1, June 2004.

[SFT01] D. Sridhara, T. Fuja, and R.M. Tanner. Low density parity check codes from permutation matrices. *Proc. Conf. on Inform. Sciences and Systems, Baltimore, MD*, March 2001.

[Sha48] C. E. Shannon. A mathematical theory of communication. In *from The Bell System Technical Journal*, volume 27, pages 379–423, 623–656, July, October 1948.

[SL08] E. Sharon and S. Litsyn. Constructing ldpc codes by error minimization progressive edge growth. *Communications, IEEE Transactions on*, 56(3) :359–368, March 2008.

[SS94] M. Sipser and D.A. Spielman. Expander codes. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 566–576, Nov 1994.

[SV06] M. Schwartz and A. Vardy. On the stopping distance and the stopping redundancy of codes. *Information Theory, IEEE Transactions on*, 52(3) :922–932, March 2006.

[SV12] R. Smarandache and P.O. Vontobel. Quasi-cyclic ldpc codes : Influence of proto- and tanner-graph structure on minimum hamming distance upper bounds. *Information Theory, IEEE Transactions on*, 58(2) :585–607, Feb 2012.

[SZ10] C. Schlegel and Shuai Zhang. On the dynamics of the error floor behavior in (regular) ldpc codes. *Information Theory, IEEE Transactions on*, 56(7) :3248–3264, July 2010.

[TAD04] J. Thorpe, K. Andrews, and S. Dolinar. Methodologies for designing ldpc codes using protographs and circulants. In *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*, pages 238–, June 2004.

[Tan81] R.M. Tanner. A recursive approach to low complexity codes. *Information Theory, IEEE Transactions on*, 27(5) :533–547, Sep 1981.

[Tan99] R. M. Tanner. On quasi-cyclic repeat-accumulate codes. *Proc. 37th Annu. Allerton Conf. Commun., Control Comput.*, pages 249–259, 1999.

[Tan01a] R.M. Tanner. Minimum-distance bounds by graph analysis. *Information Theory, IEEE Transactions on*, 47(2) :808–821, Feb 2001.

[Tan01b] R.M. Tanner. Spectral graphs for quasi-cyclic ldpc codes. In *Information Theory, 2001. Proceedings. 2001 IEEE International Symposium on*, pages 226–, 2001.

[Tho03] Jeremy Thorpe. Low-density parity-check (ldpc) codes constructed from protographs. *IPN progress report*, 42(154) :42–154, 2003.

[TJVW03] Tao Tian, C. Jones, J.D. Villasenor, and R.D. Wesel. Construction of irregular ldpc codes with low error floors. In *Communications, 2003. ICC '03. IEEE International Conference on*, volume 5, pages 3125–3129 vol.5, May 2003.

[TJVW04] Tao Tian, C.R. Jones, J.D. Villasenor, and R.D. Wesel. Selective avoidance of cycles in irregular ldpc code construction. *Communications, IEEE Transactions on*, 52(8) :1242–1247, Aug 2004.

[TKLK07] Xiongfei Tao, Jong-man Kim, Weizhong Liu, and Li Kong. Improved construction of low-density parity-check codes based on lattices. In *Information Technology Convergence, 2007. ISITC 2007. International Symposium on*, pages 208–212, Nov 2007.

[TSF01] R.M. Tanner, D. Sridhara, and T.E. Fuja. A class of group-structured ldpc codes. *Proc. of International Symposium on Communication Theory and Applications*, 52(2) :365–370, July 2001.

[TSO] Trapping set ontology, [online]. available : http ://www.ece.arizona.edu/ vasiclab/projects/codingtheory/trappingsetontology.html.

[TSS+04] R.M. Tanner, D. Sridhara, A. Sridharan, T.E. Fuja, and D.J. Costello. Ldpc block and convolutional codes based on circulant matrices. *Information Theory, IEEE Transactions on*, 50(12) :2966–2984, Dec 2004.

[TXK+04] H. Tang, Jun Xu, Yu Kou, Shu Lin, and K. Abdel-Ghaffar. On algebraic construction of gallager and circulant low-density parity-check codes. *Information Theory, IEEE Transactions on*, 50(6) :1269–1279, June 2004.

[UHLS11] A.G.D. Uchoa, C. Healy, R.C. de Lamare, and R.D. Souza. Design of ldpc codes based on progressive edge growth techniques for block fading channels. *Communications Letters, IEEE*, 15(11) :1221–1223, November 2011.

[VCNP09] B. Vasic, S.K. Chilappagari, D.V. Nguyen, and S.K. Planjery. Trapping set ontology. In *Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on*, pages 1–7, Sept 2009.

[VCSR06] B. Vasic, S. K. Chilappagari, S. Sankaranarayanan, and R. Radhakrishnan. Failures of the gallager b decoder : analysis and applications. In *in Proc. 2nd Information Theory and Applications Workshop*, Feb. 2006.

[VDP08] A. Venkiah, D. Declercq, and C. Poulliat. Randomized progressive edge-growth (randpeg). In *Turbo Codes and Related Topics, 2008 5th International Symposium on*, pages 283–287, Sept 2008.

[VDS07] D. Vukobratovic, A. Djurendic, and V. Senk. Ace spectrum of ldpc codes and generalized ace design. In *Communications, 2007. ICC '07. IEEE International Conference on*, pages 665–670, June 2007.

[VL02] P.O. Vontobel and H.A. Loeliger. Irregular codes from regular graphs. In *Information Theory, 2002. Proceedings. 2002 IEEE International Symposium on*, pages 284–, 2002.

[VM04] B. Vasic and O. Milenkovic. Combinatorial constructions of low-density parity-check codes for iterative decoding. *Information Theory, IEEE Transactions on*, 50(6) :1156–1176, June 2004.

[VPI04] B. Vasic, K. Pedagani, and M. Ivkovic. High-rate girth-eight low-density parity-check codes on rectangular integer lattices. *Communications, IEEE Transactions on*, 52(8) :1248–1252, Aug 2004.

[VS08] D. Vukobratovic and V. Senk. Generalized ace constrained progressive edge-growth ldpc code design. *Communications Letters, IEEE*, 12(1) :32–34, January 2008.

[VS09] D. Vukobratovic and V. Senk. Transactions papers evaluation and design of irregular ldpc codes using ace spectrum. *Communications, IEEE Transactions on*, 57(8) :2272–2279, Aug 2009.

[VT01] P.O. Vontobel and R.M. Tanner. Construction of codes based on finite generalized quadrangles for iterative decoding. In *Information Theory, 2001. Proceedings. 2001 IEEE International Symposium on*, pages 223–, 2001.

[WDW11a] Jiadong Wang, L. Dolecek, and R. Wesel. Controlling ldpc absorbing sets via the null space of the cycle consistency matrix. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–6, June 2011.

[WDW11b] Jiadong Wang, L. Dolecek, and R. Wesel. Ldpc absorbing sets, the null space of the cycle consistency matrix, and tanner's constructions. In *Information Theory and Applications Workshop (ITA), 2011*, pages 1–5, Feb 2011.

[WDW13] Jiadong Wang, L. Dolecek, and R.D. Wesel. The cycle consistency matrix approach to absorbing sets in separable circulant-based ldpc codes. *Information Theory, IEEE Transactions on*, 59(4) :2293–2314, April 2013.

[WDY13] Yige Wang, S.C. Draper, and J.S. Yedidia. Hierarchical and high-girth qc ldpc codes. *Information Theory, IEEE Transactions on*, 59(7) :4553–4583, July 2013.

[WDZW11] Jiadong Wang, L. Dolecek, Zhengya Zhang, and R. Wesel. Absorbing set spectrum approach for practical code design. In *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, pages 2726–2730, July 2011.

[Wib96] N. Wiberg. Codes and decoding on general graphs. In *Ph.D. dissertation, Linkoping University*, 1996.

[WYD08] Yige Wang, J.S. Yedidia, and S.C. Draper. Construction of high-girth qc-ldpc codes. In *Turbo Codes and Related Topics, 2008 5th International Symposium on*, pages 180–185, Sept 2008.

[WYZ08] Xiaofu Wu, Xiaohu You, and Chunming Zhao. A necessary and sufficient condition for determining the girth of quasi-cyclic ldpc codes. *Communications, IEEE Transactions on*, 56(6) :854–857, June 2008.

[XB04] Hua Xiao and A.H. Banihashemi. Improved progressive-edge-growth (peg) construction of irregular ldpc codes. *Communications Letters, IEEE*, 8(12) :715–717, Dec 2004.

[XCD+07] Jun Xu, Lei Chen, I. Djurdjevic, Shu Lin, and K. Abdel-Ghaffar. Construction of regular and irregular ldpc codes : Geometry decomposition and masking. *Information Theory, IEEE Transactions on*, 53(1) :121–134, Jan 2007.

[XZKB10] H. Xinde, L. Zongwang, B. V. K. V. Kumar, and R. Barndt. Error floor estimation of long ldpc codes on magnetic recording channels. *Mag. IEEE Trans.*, 46(6) :1836–1839, June 2010.

[YB04] M. Yazdani and A.H. Banihashemi. On construction of rate-compatible low-density parity-check codes. In *Communications, 2004 IEEE International Conference on*, volume 1, pages 430–434, June 2004.

[YH03] Kyeongcheol Yang and T. Helleseth. On the minimum distance of array codes as ldpc codes. *Information Theory, IEEE Transactions on*, 49(12) :3268–3271, Dec 2003.

[ZLT10] Xia Zheng, F.C.M. Lau, and C.K. Tse. Constructing short-length irregular ldpc codes with low error floor. *Communications, IEEE Transactions on*, 58(10) :2823–2834, October 2010.

[ZMZP08] Fan Zhang, Xuehong Mao, Wuyang Zhou, and H.D. Pfister. Girth-10 ldpc codes based on 3-d cyclic lattices. *Vehicular Technology, IEEE Transactions on*, 57(2) :1049–1060, March 2008.

[ZSW12a] Guohua Zhang, Rong Sun, and Xinmei Wang. Explicit construction of girth-eight qc-ldpc codes and its application in crt method. *J. on Commun.*, 33(3) :171–176, 2012.

[ZSW12b] Guohua Zhang, Rong Sun, and Xinmei Wang. New quasi-cyclic ldpc codes with girth at least eight based on sidon sequences. In *Turbo Codes and Iterative Information Processing (ISTC), 2012 7th International Symposium on*, pages 31–35, Aug 2012.

[ZSW13a] Guohua Zhang, Rong Sun, and Xinmei Wang. Deterministic construction of girth-eight (3,l) qc-ldpc codes from quadratic function. *Electronics Letters*, 49(9) :600–602, April 2013.

[ZSW13b] Guohua Zhang, Rong Sun, and Xinmei Wang. Several explicit constructions for (3,l) qc-ldpc codes with girth at least eight. *Communications Letters, IEEE*, 17(9) :1822–1825, September 2013.

[ZW10] Guohua Zhang and Xinmei Wang. Girth-12 quasi-cyclic LDPC codes with consecutive lengths. *CoRR*, abs/1001.3916, 2010.

[ZXMZ05] Fan Zhang, Ying Xu, Xuehong Mao, and Wuyang Zhou. High girth ldpc codes construction based on combinatorial design. In *Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st*, volume 1, pages 591–594 Vol. 1, May 2005.

[ZZ14] Jianhua Zhang and Guohua Zhang. Deterministic girth-eight qc-ldpc codes with large column weight. *Communications Letters, IEEE*, 18(4) :656–659, April 2014.

[ZZPY07] Wei Zhan, Guangxi Zhu, Li Peng, and Xi Yan. Quasi-cyclic ldpc codes based on d and q matrices through progressive edge growth. In *Intelligent Signal Processing and Communication Systems, 2007. ISPACS 2007. International Symposium on*, pages 12–15, Nov 2007.