Distributed coding and computing for networks

Fanny Jardel

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2016-ENST-0001

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

Télécom-ParisTech

Spécialité « Communications et Électronique »

présentée et soutenue publiquement par

Fanny JARDEL

le 11 Janvier 2016

Calcul et Stockage Distribués pour les Réseaux de Communication

Directeurs de thèse : **Joseph J. BOUTROS**
Ghaya REKAYA-BEN OTHMAN
Co-encadrement de la thèse : **Mireille SARKISS**

M. Jérôme LACAN, Professeur, ISAE	Rapporteur
M. Ramesh PYNDIAH, Directeur Scientifique, Télécom Bretagne	Rapporteur
M. Daniel AUGOT, Directeur de recherche, INRIA Saclay & LIX	Examinateur
M. Jean-Claude BELFIORE, Professeur, Télécom-ParisTech	Examinateur
M. Nicolas SENDRIER, Directeur de recherche, INRIA Rocquencourt	Examinateur
M. Rüdiger URBANKE, Professeur Ordinaire, EPFL	Examinateur
Mme. Mireille SARKISS, Docteur, CEA Saclay	Encadrant de thèse
M. Joseph J. BOUTROS, Professeur, Texas A&M at Qatar	Directeur de thèse
Mme. Ghaya REKAYA-BEN OTHMAN, Professeur, Télécom-ParisTech	Co-directeur de thèse

T H È S E

Télécom-ParisTech
Grande école de l'Institut Mines-Télécom - membre fondateur de ParisTech
46 rue Barrault 75013 Paris - (+33) 1 45 81 77 77 - www.telecom-paristech.fr

♡ ♡ ♡

To my mother and my father

To my sister and my brother

♡ ♡ ♡

Acknowledgments

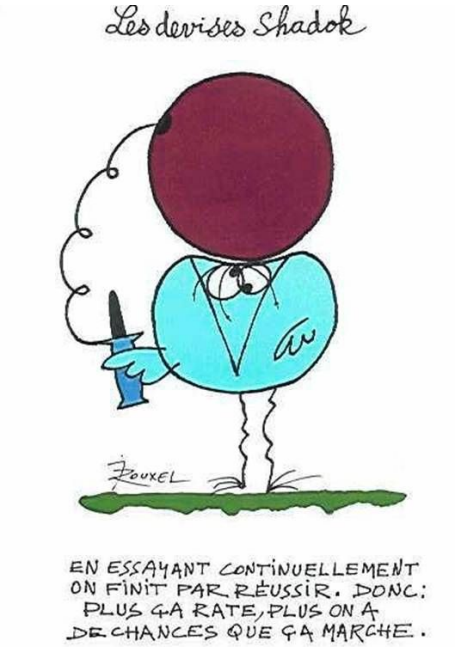
A l'issue de la rédaction de cette recherche, je suis convaincue que la thèse est loin d'être un travail solitaire. En effet, je n'aurais jamais pu réaliser ce travail doctoral sans le soutien d'un grand nombre de personnes dont la générosité, la bonne humeur et l'intérêt manifesté à l'égard de ma recherche m'ont permis de progresser dans cette phase délicate de l'apprenti-chercheur. En premier lieu, je tiens à remercier mon directeur de thèse, le Professeur Joseph J. Boutros, pour la confiance qu'il m'a accordé en acceptant d'encadrer ce travail doctoral. Sa capacité d'analyse et son enthousiasme m'ont montré que le monde de la recherche pouvait être un univers passionnant. Il a su, par sa fine maîtrise de l'encadrement, sa grande expertise professionnelle, ses multiples conseils, son expérience et son humanité, me transmettre les fondamentaux qui font au fur et à mesure d'un doctorant, un docteur. Je souhaite également remercier le Docteur Mireille Sarkiss, qui a encadré cette thèse. J'aimerais lui dire à quel point j'ai apprécié sa grande disponibilité et ses conseils scientifiques. J'ai aussi été extrêmement sensible à ses qualités humaines d'écoute et de compréhension tout au long de ce travail doctoral. Je remercie aussi le Professeur Ghaya Rekaya pour m'avoir donné l'opportunité et l'envie de commencer ces travaux de recherches, ainsi que pour toutes nos discussions et ses conseils qui m'ont accompagnée tout au long de mon cursus.
Également, je tiens à remercier tous les membres de mon jury de thèse. Je remercie tout particulièrement le Professeur Jérôme Lacan et le Professeur Ramesh Pyndiah qui, en leur qualité de rapporteurs de ma thèse, sont les évaluateurs de ce travail de recherche. Je remercie les Professeurs Daniel Augot, Jean-Claude Belfiore, Nicolas Sendrier et Rüdiger Urbanke d'avoir participé à ce jury de thèse en tant qu'examinateurs. Leurs expertises seront aussi très précieuses pour m'aider à ouvrir de nouvelles perspectives dans la suite de mon travail de recherche.

Cette thèse est le résultat de trois années de travail passées au CEA et à Télécom ParisTech. J'adresse toute ma gratitude à Christophe Janneteau pour m'avoir accueillie au sein du laboratoire LSC, ainsi qu'à tous mes collègues avec qui j'ai pu échanger et auprès de qui je me suis enrichie. Je souhaite remercier tous ceux qui, de près ou de loin,

ont participé à l'élaboration de cette thèse. J'adresse une pensée particulière à tous mes amis sans qui ces trois années n'auraient pas été aussi joyeuses, Marco, Alexis, Sofiane, Sarah, Raphaël, Sarah, Matha, Radhouène, Pierrick, Loic, Goeffrey, Nicola, Patrick, Asma, Elie, Antonio et tous les autres, merci pour tous les moments, discussions et fous rires que nous avons partagé et pour votre support sans faille.

Mes pensées vont maintenant vers ma famille, mes grands-parents qui m'ont toujours soutenue et poussée vers l'avant. Ma sœur Marie et mon frère Camille qui ont été pour moi une source de motivation, optimisme et énergie positive. Mes deux parents, sans qui, rien n'était possible. Qu'ils sachent que leur amour et leur confiance m'ont élevée jusque là. Et sans oublier, l'amour de ma vie.

Je tourne une page et j'en ouvre une autre: vivement la suite!



Abstract

Cloud storage and computing is a new area of application of coding theory. In this young topic, models are yet simple and consider only erasures for error correction. The erasure model is a powerful algebraic tool to understand some intrinsic properties of codes without dealing with complicated channels such those including errors or memory. In distributed storage, the erasure model is combined to special system constraints that did not previously exist. This thesis on coding for erasures is divided into two parts.

The first part of this work is devoted to performance study of spatially coupled codes for erasure channels. A new method for spatial coupling of low-density parity-check ensembles (LDPC) is proposed. The method is inspired from overlapped layered coding. Edges of local ensembles and those defining the spatial coupling are separately built. The new method allows the construction of non-uniform coupling chains with near-Shannon spatially-varying thresholds under iterative decoding. However, random LDPC ensembles do not achieve full diversity on non-ergodic channels. To cope with block erasures and quasi-static fading, LDPC ensembles with a special structure are considered. These codes are known as Root-LDPC in the literature. A Root-LDPC ensemble guarantees that any information bit receives messages from all channel states. Results in the literature show that the gap between the Root-LDPC boundary and the capacity boundary in the fading plane is not small enough. We propose to saturate the whole Root-LDPC boundary via spatial coupling. For simplicity, we adopt an equivalent erasure channel model rather than a block fading model. It is shown that spatial coupling of parity bits only is sufficient to saturate the Root-LDPC threshold boundary in the erasure plane. Then, spatial coupling is applied to parity bits of a Root-LDPC ensemble designed for a channel with 4 block-erasure states and a maximal design rate of 3/4 attaining double diversity. The advantage of spatial coupling is shown in the erasure plane as an improvement of a threshold boundary, under independent erasures. The spatial coupling maintains the double diversity because it connects parity bits only. The drawback of this partial coupling is a weak saturation of the threshold boundary towards the capacity boundary.

In the second part of this work, we consider non-binary product codes with MDS components and their iterative row-column algebraic decoding on the erasure channel. Both independent and block erasures are considered. A compact graph representation is introduced on which we define double-diversity edge colorings via the rootcheck concept. An upper bound of the number of decoding iterations is given as a function of the graph size and the color palette size M. Stopping sets are defined in the context of MDS components and a relationship is established with the graph representation. A full characterization of these stopping sets is given up to a size $(d_1 + 1)(d_2 + 1)$, where d_1 and d_2 are the minimum Hamming distances of the column and row MDS components respectively. Then, we propose a differential evolution edge coloring algorithm that produces colorings with a large population of minimal rootcheck order symbols. The complexity of this algorithm per iteration is $o(M^{\aleph})$, for a given differential evolution parameter $\aleph$, where $M^{\aleph}$ itself is small with respect to the huge cardinality of the coloring ensemble. The performance of MDS-based product codes with and without double-diversity coloring is analyzed in presence of both block and independent erasures. In the latter case, ML and iterative decoding are proven to coincide at small channel erasure probability. Furthermore, numerical results show excellent performance in presence of unequal erasure probability due to double-diversity colorings.

Contents

List of Figures

List of Tables

List of Abbreviations

The abbreviations used in this work are summarized in the following.

AWGN	Additive White Gaussian Noise
BEC	Binary Erasure Channel
BMS	Binary Memoryless Symmetric
BSC	Binary Symmetric Channel
CEC	Color Erasure Channel
SEC	Symbol Erasure Channel
BP	Belief Propagation
MAP	Maximum A Posteriori
ML	Maximum Likelihood
SISO	Soft-Input Soft-Output
APP	A Posteriori Probability
DE	Density Evolution
EXIT	Extrinsic Information Transfer
GEXIT	Generalized Extrinsic Information Transfer
LLR	Log Likelihood Ratio
PCE	Parity Check Equation
SPC	Single Parity Check
UEP	Unequal Error protection
BCH	Bose-Chaudhuri-Hochquenghem
GLD	Generalized Low Density
LDPC	Low-Density Parity-Check
LT	Luby Transform
MDS	Maximum Distance Separable
RM	Reed-Muller
RS	Reed-Solomon

HDFS	Hadoop Distributed File System
RAID	Redundant Array of Independent Disks
DECA	Differential Edge Coloring Algorithm
MBR	Minimum Bandwidth Regenerating
MSR	Minimum Storage Regenerating

List of Notations

We consider in this work the following notations:
Vectors are written in boldface and lowercase.

$\mathbb{F}_q$	Finite field of size q
$\mathbb{R}$	The field of real numbers
$\mathbb{N}$	The set of all natural numbers $0, 1, 2, 3, \ldots$
log	The logarithm operation to the base 2
ln	The natural logarithm
$\lfloor x \rfloor$	Largest integer not greater than x
$\lceil x \rceil$	Smallest integer not less than x
$\lvert E \rvert$	Cardinality of the set E
$\otimes$	Kronecker product
$o(x)$	Bachmann–Landau small o notation
$O(x)$	Bachmann–Landau big 0 notation
$\mathbb{1}_{\{x=a\}}$	Indicator function
$\propto$	Proportional to
$\#$	Number of
$]a, b[$	$\{x \in \mathbb{R} : a < x < b\}$

Résumé Détaillé de la Thèse

Les technologies électroniques et logicielles actuelles combinées aux outils mathématiques et motivées par la théorie du codage, la théorie de l'information et de la communication sont en train de radicalement changer notre société. Ces outils ont aidé les technologies des réseaux sans fil à faire des progrès spectaculaires et à devenir une composante presque omniprésente de la vie moderne en révolutionnant l'éducation, la culture et l'échange d'information en général. De nouvelles techniques très efficaces sont apparues récemment pour les communications point à point et pour la transmission et le traitement de l'information au sein d'un réseau, à savoir: le codage de réseaux, le codage pour le stockage distribué, les codes basés sur les graphes spatialement couplés et les réseaux de points de grandes dimensions pour le décodage itératif.

De nos jours, au sein des systèmes d'information, le Cloud Computing est un concept commun où la plupart des ressources sont disponibles en ligne. L'intérêt du cloud computing réside dans le fait que les ressources évoluent d'une manière instantanée et automatique. Il existe dans les systèmes de communication une sous-classe du cloud computing connue sous le nom de calcul distribué. Dans ce cas particulier, plusieurs nœuds du réseau travaillent conjointement pour résoudre un problème ou pour répondre à une demande. Le stockage distribué est aussi un cas particulier du calcul distribué, où un fichier est stocké en fragments sur plusieurs nœuds du réseau.

Cette thèse est dédiée à l'étude du stockage distribué dans les centres de données en interaction avec les ressources du cloud computing, au service des clients ou des utilisateurs mobiles. Un scénario type est illustré Figure 1 où une source, demandant les services du cloud, disperse un fichier sur n nœuds. Les codes correcteurs d'erreurs sont les outils principaux utilisés dans le contexte du calcul/stockage distribué. Outre le trivial codage à répétition, un code Reed-Solomon ou tout autre code code à distance maximale séparable (MDS) est capable de reconstruire le fichier source [54]. L'objectif principal du codage distribué pour le stockage est d'augmenter la fiabilité de la reconstruction et de minimiser la bande passante requise pour cette reconstruction.

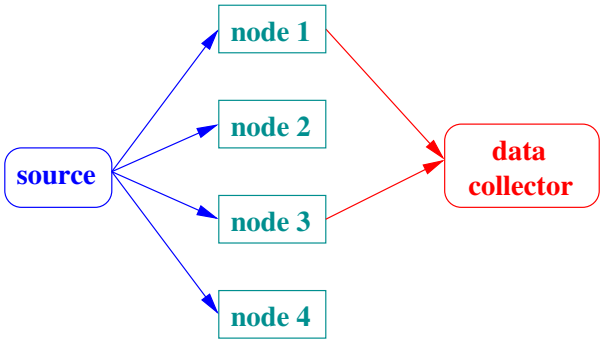


Figure 1: Système de stockage distribué d'un fichier réparti sur $n = 4$ nœuds et reconstruit à partir de $k = 2$ nœuds.

Le codage peut être utilisé pour offrir différents niveaux de protection contre les erreurs tout en assurant la qualité de service et les contraintes de puissance requises. Ces contraintes dépendent de l'état du canal sans fil entre les k nœuds et le terminal. De tels schémas de codage pour protéger de façon inégale la donnée contre les erreurs (UEP) ont été étudiés pour les réseaux sans fil et sont basés sur les codes de contrôle de parité à faible densité (LDPC), voir par exemple [73]. D'autre part, en cas de défaillance de nœuds au sein du réseau, le codage peut également être utilisé pour régénérer ces nœuds perdus. Au cours du processus de réparation, le nœud de remplacement peut contenir les mêmes informations ou il peut contenir une information équivalente. La bande passante requise pour ce processus de réparation doit être minimisée, c'est-à-dire réduire au minimum le nombre de téléchargements à effectuer au sein du réseau pour remplacer le nœud défaillant.

De nombreux codes sont des candidats potentiels pour le stockage distribué, la plupart d'entre eux sont des codes concaténés. Nous considérerons dans cette thèse les codes LDPC et les codes produits. Un code LDPC est la concaténation d'un grand nombre de petits nœuds de contrôle. Nous nous intéresserons plus particulièrement au couplage spatial des codes LDPC, impliquant de multiples copies du même ensemble LDPC. La connexion entre ces différentes copies est effectuée par des arêtes de couplage. Un code produit, quant à lui, est un code bidimensionnel construit à partir de codes élémentaires lignes et colonnes. Sa représentation matricielle équivaut à un graphe bipartite complet. Les codes produits sont également des codes concaténés, ils peuvent être considérés comme des cousins des codes LDPC de par leur décodage itératif ligne-colonne.

Ce résumé détaillé est composé de quatre parties principales. Le contenu et les contributions de chaque partie sont résumés ci-dessous.

Le premier chapitre est dédié à l'étude des codes existant pour les systèmes de stockage distribué. Dans un premier temps, nous étudierons le cas particulier des centres de données Facebook afin de pouvoir mettre en avant les principales contraintes liées au

stockage distribué. Dans les systèmes de stockage actuels, le codage des données est effectué via deux approches: la réplication et les codes à effacement MDS. Ces deux méthodes seront analysées et discutées dans ce chapitre.
Dans la littérature, d'autres techniques de codage ont été proposées pour remédier aux inconvénients du processus de réparation des codes à effacement MDS, plus précisément les codes à régénération et les codes localement réparables. Dans le contexte du stockage distribué, les codes basés sur les graphes à faible densité, tels que les codes Fontaine et les codes LDPC, peuvent également être utilisés après quelques arrangements pratiques. Une étude de ces différents codes sera effectuée dans ce chapitre.

Dans le deuxième chapitre, nous nous focalisons sur le couplage spatial des codes LDPC et Root-LDPC. Une nouvelle méthode de couplage spatiale des ensembles LDPC sera proposée. Cette méthode est inspirée du codage par couches superposées. Les arêtes des ensembles locaux et celles définissant le couplage spatial sont construites séparément. Cette nouvelle technique permet la construction de chaînes de couplage non uniforme assurant un seuil variant en fonction de la position spatiale et dont la saturation se fait pratiquement au seuil de Shannon. Les seuils de ces chaînes de couplage s'obtiennent avec un décodage itératif. Le couplage spatialement variant est effectué en collant des ensembles LDPC de rendements différents. Le canal à effacement binaire (BEC) est le canal de transmission par défaut, cependant, tous les résultats sont extensibles aux canaux sans mémoires symétriques binaires.
Les codes LDPC n'atteignent pas une diversité pleine sur les canaux non ergodiques. Pour faire face aux effacements par blocs et aux évanouissements quasi-statiques, nous considérons dans la suite une structure spéciale appelée code Root-LDPC. Un ensemble Root-LDPC garantit que tous les bits d'information reçoivent les messages de tous les états du canal. Les résultats dans la littérature montrent que l'écart entre la frontière de coupure du code Root-LDPC et celle de la capacité n'est pas assez petit dans le plan à évanouissement. Nous proposons de saturer l'ensemble de la frontière de coupure du code Root-LDPC à l'aide du couplage spatial non uniforme. Par souci de simplicité, nous adoptons un modèle équivalent de canal à effacement par blocs plutôt qu'un modèle à évanouissements par blocs. Il est aussi démontré que le couplage spatial des bits de parité uniquement est suffisant pour saturer le seuil de coupure du code Root-LDPC dans le plan à effacement.
Dans la dernière partie de ce chapitre, nous appliquons ce couplage spatial partiel à un code Root-LDPC conçu pour un canal à effacement à quatre états, ayant un rendement de 3/4 et atteignant une double diversité. L'avantage du couplage spatial est représenté dans le plan à effacement comme une amélioration du seuil de coupure du code, sous effacements indépendants. Le couplage spatial maintient la double diversité car uniquement les bits de parité sont couplés. L'inconvénient de ce couplage partiel est une faible saturation du seuil de coupure du code à la frontière de coupure de la capacité. Cette méthode est proposée dans le cadre de la conception de schémas de codage pour les systèmes de stockage distribué.

Dans le troisième chapitre, nous considérons les codes produits non binaires à codes élémentaires MDS et leur décodage algébrique itératif ligne-colonne sur le canal à effacement. Les effacements indépendants et par blocs sont considérés dans cette partie.
Une représentation graphique compacte des codes produits est introduite sur laquelle nous définissons la notion de coloration d'arêtes à double diversité via le concept de rootcheck. Une borne supérieure du nombre d'itérations de décodage est donnée en fonction de la taille du graphe et de la taille M de la palette de couleurs considérées. Une couleur correspond, dans notre modèle, à un état du canal.
Les ensembles d'arrêt sont définis dans le contexte de codes produits à codes élémentaires MDS et une relation est établie avec la représentation graphique. Une caractérisation complète de ces ensembles d'arrêt est donnée jusqu'à une taille $(d_1 + 1)(d_2 + 1)$, où d_1 et d_2 sont respectivement les distances de Hamming minimales des codes élémentaires MDS lignes et colonnes.
Un algorithme de coloration d'arêtes à évolution différentielle est ensuite proposé. Cet algorithme produit des colorations du graphe compact avec un grand nombre de symboles rootcheck d'ordre minimal. La complexité de cet algorithme par itération est $o(M^{\aleph})$, pour un paramètre d'évolution différentielle $\aleph$ donné, où $M^{\aleph}$ est petit par rapport au grand cardinal de l'ensemble de colorations possibles sur le graphe.
Les performances des codes produits à codes élémentaires MDS avec et sans coloration à double diversité sont analysées en présence d'effacements par blocs et d'effacements indépendants. Dans ce dernier cas, nous avons prouvé que les performances des décodeurs ML et itératif coïncident à faible probabilité d'effacement. De plus, les résultats numériques montrent une excellente performance en présence de effacements à probabilité inégale en raison de la coloration à double diversité.

Enfin, les résultats de ces travaux et les futures perspectives de recherche sont résumés dans une conclusion générale.

Chapitre 1: Codage pour le stockage distribué

Les codes correcteurs d'erreurs sont un des outils les plus utilisés pour le calcul et le stockage distribué. Outre la technique triviale du codage à répétition, un code Reed-Solomon ou tout autre code MDS, est capable de reconstruire le fichier source [54]. Plus récemment, de nouveaux schémas de codage concus pour les systèmes de stockage distribué ont été développés [29][25][70]. Nous visons dans ce chapitre à présenter les contraintes du stockage distribué et un aperçu des codes existants.

Paramètres et contraintes du stockage distribué

Dans cette thèse, seuls les réseaux de stockage avec un grand nombre de serveurs sont considérés. Avant de définir les enjeux et paramètres des systèmes de stockage distribué, nous allons étudier le cas particulier d'un centre de données Facebook. Ce cas pratique nous permettra de mieux comprendre et cibler les enjeux du stockage distribué. Une

description complète est disponible dans [80].
Un cluster de stockage est divisé en deux clusters Hadoop (HDFS). Hadoop est un framework libre et open source écrit en Java, destiné à faciliter la création d'applications distribuées et échelonnables, permettant aux applications de travailler avec des milliers de nœuds et des pétaoctets de données. Ainsi, chaque nœud est constitué de machines standards regroupées en grappes (clusters). Dans l'exemple d'un centre de données Facebook, chaque appareil stocke jusqu'à 36 téraoctets de données. Les données les plus fréquemment consultées sont stockées sous la forme de trois répliques. Les données qui n'ont pas été consultées pendant plus de trois mois sont codées avec un code Reed-Solomon [14, 10]. Les différents blocs d'un mot de code sont répartis aléatoirement sur les machines. Dans la suite, une machine peut aussi être désignée comme un nœud du réseau de stockage. Au sein du centre de données, une médiane de 50 machines sont indisponibles pour plus de 15 minutes par jour. Lorsque les blocs sont manquants dans un mot de code, en moyenne 98.08% ont exactement un bloc manquant, 1.87% ont deux blocs manquants et seulement 0.05% ont trois ou plusieurs blocs manquants.
Basé sur l'étude des clusters Facebook, nous pouvons résumer les cinq points principaux de la conception de bons codes pour les systèmes de stockage distribué.

- Codage. Cette contrainte est directement liée à l'optimisation de l'espace de stockage dans les centres de données. Le service vendu par le fournisseur de cloud à un utilisateur final est un niveau de fiabilité. L'un des principaux défis pour les fournisseurs de cloud est de faire un compromis entre un niveau de fiabilité proposé au client et l'espace de stockage dédié aux données. En effet, le niveau de la fiabilité dépend de la quantité de redondance dans la structure du code alors que l'espace de stockage dans un cluster doit être optimisé. Par conséquent, le choix des paramètres du code, définissant son rendement, est crucial. De plus, un codeur systématique est plus approprié aux systèmes de stockage distribué. En effet, le temps nécessaire pour lire les données d'information est ainsi minimisé.

- Allocation. Le problème de l'allocation peut être défini en une question. Comment distribuer les symboles codés à travers le réseau afin d'être robuste face aux effacements?

- Mise à jour. La structure du code doit éviter un ré-encodage complet des données lors de leur mise à jour. Certaines applications ne nécessitent pas de mettre à jour les données une fois stockées.

- Reconstruction. Dans un système de stockage distribué, lorsqu'une défaillance de nœuds se produit, l'objectif principal est de reconstruire ces nœuds défaillants de manière à ce que le client/collecteur soit toujours en mesure de récupérer ses données, en communiquant avec un sous-ensemble des nœuds du réseau. Deux méthodes différentes existent pour régénérer un nœud défaillant. Lorsque le nœud défaillant est régénéré à l'identique, le processus est appelé la réparation exacte. Toutefois, cette contrainte peut être relaxée: la réparation est dite fonctionnelle. Dans ce cas, les blocs régénérés peuvent contenir des données différentes de celles

initialement contenues dans nœud défaillant. Toutefois, le système réparé doit conserver les propriétés du code. De l'étude du cas Facebook, nous pouvons déduire que la défaillance d'un nœud est le scénario le plus courant.

- Le téléchargement de données. Pour les applications de stockage distribué, le choix de la méthode de décodage est un autre challenge critique. La conception du code doit permettre un processus de décodage itératif, afin d'éviter la grande complexité du décodage à maximum de vraisemblance (ML). Avec un décodeur ML, tous les nœuds doivent être contactés afin de réparer le nœud défaillant. Il en résulte donc une grande complexité de décodage et une consommation élevée de la bande passante au sein du réseau.

Le principal objectif du codage pour le stockage est d'augmenter la fiabilité de la reconstruction des données, tout en réduisant la bande passante de réparation requise et le nombre de nœuds à contacter par réparation. En effet, le trafic au sein du réseau ne doit pas être accaparé par le trafic inter-rack en raison des opérations de reconstruction. Les codes optimisant cette contrainte en bande passante sont appelés les codes à régénération (regenerating codes) [28].
Dans un même temps, au cours d'un processus de réparation, le nombre de nœuds à être contactés doit être réduit, c'est-à-dire le code doit être à localité faible. Le concept de code à localité a été défini par Gopalan et al. comme la localité de l'information [40]. La localité d'un code est donnée par la définition ci-dessous dans le contexte classique d'effacements indépendants. Considérons un code linéaire $C[n, k, d]_q$. Un mot de code de C est écrit comme un vecteur à n symboles $(c_1, c_2, \ldots, c_n)$. La localité de l'information, $Loc(C)$, de C est définie comme:

$$Loc(C) = max_i \; Loc(c_i),$$

où un symbole du code c_i a une localité $Loc(c_i) = r$ si il peut être reconstruit en accédant seulement à r autres symboles, i.e. $c_i = \sum_{j=1}^{r} \lambda_j c_j$, où $\lambda_j \in F_q$.
La localité d'un symbole est le nombre d'autres symboles à contacter dans le pire cas afin de réparer un symbole effacé. Une réparation rapide est assurée par une faible localité des symboles. Une nouvelle classe de codes minimisant la localité est apparue dans la littérature. Ces codes sont appelés codes localement réparables (Locally Repairable Codes) et ont été introduits dans [49] et développés dans [119], [3] et [70].

Toutes les contraintes du codage pour le stockage distribué ont été présentées. Les différents schémas de codage mis en œuvre dans les centres de données actuels sont maintenant examinés.

Les méthodes d'aujourd'hui: réplication et codes à effacement

De nos jours, le codage pour les systèmes de stockage distribué est réalisé en utilisant principalement deux approches: les codes à réplication et les codes à effacement. La

réplication, qui est la méthode la plus largement utilisée, consiste à stocker une copie de la totalité des données sur plusieurs nœuds de stockage. En pratique, les données des utilisateurs sont répliquées/copiées sur trois nœuds de stockage différents. Ceci permet de restaurer facilement les données en cas de défaillance d'un nœud. En effet, les données sont récupérées à partir de l'un des deux autres nœuds disponibles et stockées sur un nouveau nœud sain. Du point de vue du codage, la réplication est un code à répétition. Cette technique est simple à mettre en œuvre, mais est inefficace en termes d'optimisation de l'espace de stockage.

Afin de remédier à ce problème, une approche utilisant les codes à effacement MDS a été mise en place dans certains systèmes de stockage tels que Facebook. Dans ce cas, les données initiales sont divisées en k morceaux et de la redondance est ajoutée pour créer n blocs codés, qui sont ensuite distribués sur n nœuds du réseau. Ainsi, chaque nœud stocke seulement une partie des données codées. Cela permet à un utilisateur final de reconstruire l'ensemble des données en téléchargeant des morceaux codés de tout k (ou plus) nœud voisin. Quand un nœud est effacé, il est remplacé par un nouveau nœud vide qui télécharge l'ensemble des données de ses k voisins pour reconstruire l'information stockée avant la panne. Par conséquent, au moins k nœuds doivent être accessibles à tout moment.
Les codes à effacement sont inefficaces dans le sens où le nœud de remplacement doit reconstruire la donnée entière pour finalement n'en stocker qu'une petite partie. Ils souffrent d'un processus de réparation très lent, en particulier dans le cas de la réparation d'un seul nœud, ce qui est le scénario le plus commun. Néanmoins, les codes à effacement fournissent un meilleur compromis fiabilité-redondance que les codes à répétition. Par conséquent, une tâche importante dans la conception des codes à effacement est de réaliser une réparation rapide et de pouyoir supporter un grand nombre de pertes.

En dépit de leurs caractéristiques mal adaptés pour les contraintes du stockage distribué, les codes à effacement MDS (les codes Reed-Solomon) sont largement utilisés. La principale raison est historique: lorsque les premiers réseaux de stockage de données sont apparus, les recherches concernant les autres codes à effacement n'étaient pas aussi avancées. De plus, les codes Reed-Solomon ont été mis en œuvre dans les systèmes RAID pendant une longue période, et pendant longtemps les différents niveaux de RAID répondaient aux besoins des utilisateurs. Avec l'émergence des grands réseaux de stockage, i.e. avec un grand nombre de serveurs/nœuds, de nouveaux systèmes de codage basés sur les codes Reed-Solomon ont été développés [91] [78] [33]. Ces codes ont été conçus afin d'améliorer la localité des codes Reed-Solomon et d'améliorer le processus de réparation des codes à effacement MDS.

Récemment, dans la littérature, d'autres techniques de codage ont été proposées pour surmonter les inconvénients du processus de réparation des codes à effacement MDS: les codes à régénération et les codes localement réparables.

Codes à régénération

Les codes à régénération ont été introduits par Dimakis et al. dans [28] comme une nouvelle classe de codes à réparation exacte. D'une manière similaire aux codes à effacement, chaque nœud stocke de petits morceaux de données codées et l'utilisateur final peut récupérer les données d'origine à partir de k nœuds. Cependant, dans cette classe de codes, en cas de défaillance d'un nœud, le nœud de remplacement est capable de récupérer les données antérieures à l'échec en téléchargeant uniquement une partie des données stockées sur $d \geq k$ nœuds et non plus l'ensemble des données stockées sur ces nœuds [79] [80] [29]. Ainsi, ils permettent une réduction drastique de la quantité de données à télécharger pour une réparation.

Dans la littérature, deux classes de codes à régénération ont émergé, principalement les codes à régénération minimisant la bande passante de réparation (les codes MBR) et la classe des codes à régénération minimisant l'espace stockage (les codes MSR). Les codes MSR sont des codes à efficacité optimale pour le stockage. Ils ont besoin d'un espace de stockage minimal par nœud. Les codes MBR sont des codes à efficacité optimale du point de vue de la bande passante. Ils ont besoin du minimum de bande passante de réparation, en d'autres termes, lors de la réparation le nœud télécharge précisément la quantité de données qu'il souhaite stocker.

Codes localement réparables

Les codes localement réparables sont une autre approche de codage qui exploite le concept de localité. Ces codes sont conçus de telle sorte que le nombre d de nœuds à contacter pour la réparation d'un nœud défaillant est plus petit que k: $d \leq k$. Gopalan et al. ont proposé dans [40] une version modifiée de la borne du Singleton pour les codes localement réparables. Considérons un code linéaire $C[n, k, d]_q$ avec une localité $Loc(C)$, la relation entre la localité du code et ses paramètres est donnée par

$$d \leq n - k - \lceil \frac{k}{Loc(C)} \rceil + 2. \tag{1}$$

Une nouvelle classe de codes localement réparables, présentée par Oggier et *al.* dans [69], comprend les codes auto-réparables (Self-repairing codes). Les codes auto-réparables ne sont pas MDS et suivent le même processus de codage que les codes à régénération. Ils minimisent le nombre de nœuds à être contactés pour réparer une défaillance. Les fragments codés peuvent être réparés directement à partir d'autres sous-ensembles de fragments codés, sans avoir à reconstruire la donnée originale entière. Le nombre de fragments codés contactés pour la réparation est fixe et ne dépend que du nombre de blocs codés manquants et non pas de quel bloc spécifique est manquant.

Conclusions

Nous pouvons maintenant énumérer les critères nécessaires à la conception d'algorithmes de réparation efficaces dans le cadre du stockage distribué:

- Le processus de réparation ne doit pas être un décodage complet. En effet, un décodage itératif permettant une réparation locale est privilégié.
- Le nombre de nœuds voisins à contacter durant le processus doit être minimisé.

Les codes décrits précédemment sont des codes en blocs. Dans le contexte du stockage distribué, les codes en graphes clairsemés, tels que les codes Fontaine, les codes Raptor et les codes LDPC, peuvent également être utilisés après quelques arrangements pratiques. En effet, une localité faible peut être obtenue via la conception de matrices de contrôle de parité creuses. Motivés par ce résultat, nous nous sommes intéressés aux codes LDPC spatialement couplés. Ces codes peuvent atteindre les performances d'un décodage (Maximum-à-posteriori) MAP avec un décodage itératif.
De plus, jusqu'à maintenant, les codes existant dans la littérature ont été développés dans le cas où seulement un petit nombre de nœuds seraient effacés. Aucun schéma de codage permettant de faire face à une perte de cluster n'a été proposé. Dans les prochains chapitres, nous utiliserons le critère de la diversité afin de concevoir de nouveaux schémas de codage permettant d'être efficace dans le cas où un cluster est effacé (en plus des effacements de symboles uniformes).

Chapitre 2: Couplage spatial pour la diversité

Dans ce chapitre, nous proposons une nouvelle méthode de couplage spatial. Notre méthode de couplage, contrairement à celle proposée dans la littérature [55], ajoute de nouvelles arêtes d'un seul côté seulement pour coupler un ensemble LDPC à son voisin. Cette méthode, appelée couplage spatial multicouches à sens direct des codes LDPC, est inspirée de la protection inégale de l'information face aux erreurs (UEP) [19][45].

Construction du couplage spatial multicouches à sens direct des codes LDPC

Le codage en couches est généralement fait par binning ou superposition [103]. Une autre méthode simple pour construire des codes en couches pour l'UEP est le chevauchement. C'est-à-dire, que la sortie du premier codeur est partiellement ou totalement appliquée à l'entrée du second codeur. Le processus est répété pour un groupe de L_c codeurs. Un exemple de codage par couches superposées est représenté sur la Figure 2.

Tout d'abord, une somme directe de L_c codes LDPC réguliers $(3, 6)$ est construite en copiant un ensemble $(3, 6)$ sur la diagonale principale de la matrice de contrôle de parité. Ensuite, chaque matrice de contrôle de parité, des ensembles LDPC $(3, 6)$ non couplés, est connectée à son voisin par une matrice creuse $(1, 2)$ (matrice binaire aléatoire creuse avec un poids de un par colonne et un poids de deux par ligne).

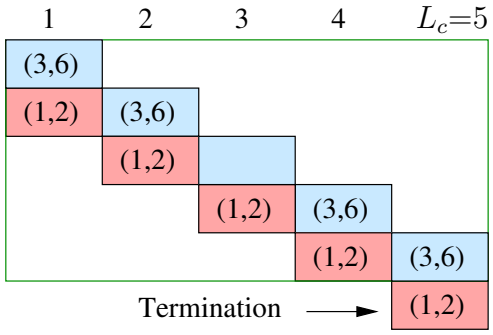


Figure 2: Matrice de contrôle de parité, d'un code multicouches à sens direct de longueur L_c, construite à partir d'ensembles LDPC réguliers $(3, 6)$ couplés via une matrice creuse binaire $(1, 2)$.

La structure finale de la matrice de contrôle de parité est très similaire à un ensemble $(4, 8, L_c, w = 2)$ couplé comme on peut le voir Figure 2; ces deux structures diffèrent dans la façon dont les arêtes connectent les ensembles voisins entre eux. Plus précisément, le couplage spatial multicouches à sens direct nous permet d'introduire un nouvel ensemble couplé non uniforme dont les paramètres sont $(d_b, d_c; d_{bs}, d_{cs}; L_c)$. Nous appelons notre structure de couplage non uniforme car elle est moins déterministe que celle proposée dans la littérature. De plus, notre méthode ne requiert pas de double moyennage sur les arêtes.

Construction d'un ensemble couplé non uniforme $(d_b, d_c; d_{bs}, d_{cs}; L_c)$**:**
Définir L_c positions spatiales sur une ligne horizontale. A chaque position spatiale i, $i = 1 \dots L_c$, on place M_b nœuds variables et $\frac{d_b}{d_c} M_b$ nœuds contraintes. Des nœuds contraintes supplémentaires peuvent être ajoutés en positions $i > L_c$ dans le but de la terminaison à droite de la chaîne couplée. Les d_b arêtes d'un nœud variable en position i sont supposées être connectées aux nœuds contraintes de la même position spatiale i. Les d_c arêtes d'un nœud contrainte en position i sont supposées être connectées aux nœuds variables en position i. Le couplage multicouches à sens direct est composé d'arêtes de couplage supplémentaires. Chaque nœud variable en position i a d_{bs} arêtes supplémentaires connectées aux nœuds contraintes en position $i + 1$. Chaque nœud contrainte en position i a d_{cs} arêtes supplémentaires connectées aux nœuds variables en position $i - 1$. L'ensemble de couplage (d_{bs}, d_{cs}) doit satisfaire la contrainte de couplage multicouches: $\frac{d_{bs}}{d_{cs}} = \frac{d_b}{d_c}$. Cette contrainte traduit le fait que les deux matrices (d_b, d_c) et (d_{bs}, d_{cs}) ont la même taille. La structure de la Figure 2 correspond à un ensemble $(3, 6; 1, 2; L_c)$. Sa représentation en graphe compact (avec protographes) est donnée Figure 3.

Sans terminaison de la chaîne de couplage, le rendement pour l'ensemble couplé $(d_b, d_c; d_{bs}, d_{cs}; L_c)$ est $1 - \frac{d_b}{d_c}$. Le couplage multicouches à sens direct n'a pas de terminaison à gauche de la chaîne. Une terminaison peut être placée à droite, i.e. un ensemble

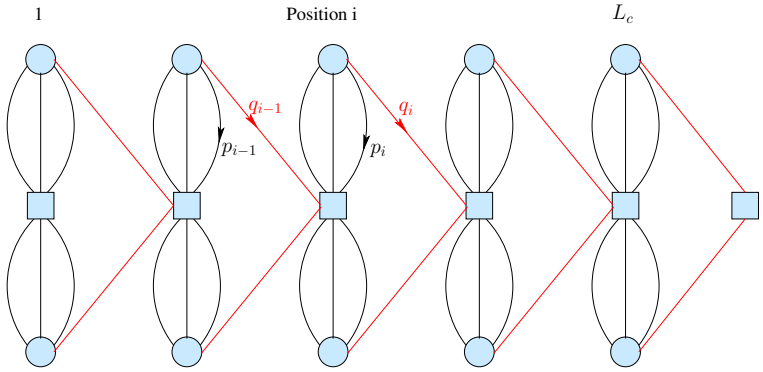


Figure 3: Graphe de Tanner du couplage spatial multicouches à sens direct d'un ensemble LDPC $(3, 6)$.

local (d_{bs}, d_{cs}) supplémentaire avec ses nœuds contraintes en position $i = L_c + 1$. La perte en rendement due à la terminaison est un facteur $1 - 1/L_c$.

Les performances de la chaîne couplée sont étudiées via les équations à évolution de densité. Le couplage spatial multicouches à sens direct d'un ensemble $(d_b, d_c; d_{bs}, d_{cs}; L_c)$ nécessite plus d'équations à évolution de densité que le couplage comme fait dans la littérature dû à sa structure à types d'arêtes multiples. Ici, le type concerne le message véhiculé par l'arête. Définissons p_i comme le message d'un nœud variable en position i à un nœud contrainte en position i et q_i le message allant d'un nœud variable en position i au nœud contrainte en position $i + 1$.

Proposition *Soit d_b, d_c, d_{bs} et d_{cs} des entiers naturels satisfaisant $\frac{d_{bs}}{d_{cs}} = \frac{d_b}{d_c}$. Les équations à évolution de densité pour un ensemble couplé multicouches à sens direct $(d_b, d_c; d_{bs}, d_{cs}; L)$ sont*

$$\begin{aligned} p_i &= \epsilon \cdot f_{i+1}^{d_{bs}} \cdot \left(1 - (1 - p_i)^{d_c - 1}(1 - q_{i-1})^{d_{cs}}\right)^{d_b - 1}, \\ q_i &= \epsilon \cdot f_{i+1}^{d_{bs} - 1} \cdot \left(1 - (1 - p_i)^{d_c - 1}(1 - q_{i-1})^{d_{cs}}\right)^{d_b}, \\ f_{i+1} &= \left(1 - (1 - p_{i+1})^{d_c}(1 - q_i)^{d_{cs} - 1}\right), \end{aligned}$$

pour $i = 1 \ldots L_c$, où f_{i+1} est un message d'un nœud contrainte en position $i + 1$ à un nœud variable position i se propageant vers le passé.

La propagation de croyance des seuils est réalisée sur un canal BEC pour différents ensembles LDPC de rendement $1/2$. Ces seuils sont listés Table 1 pour $d_{bs} = 1$ et $d_{cs} = 2$. Les performances d'un ensemble couplé $(d_b, d_c, L_c, w = 2)$ comme proposé dans la littérature (4ème colonne) sont comparées à celles d'un ensemble couplé multicouches à sens direct $(d_b - 1, d_c - 2; 1, 2; L_c)$ (5ème colonne). Pour les deux ensembles couplés, une

terminaison à droite de la chaîne est faite par $p_{L_c+1} = 0$. Pour toutes les longueurs de chaînes L_c suffisamment grandes, les résultats sur le seuil sont identiques, typiquement $L_c = 60$ ou $L_c = 100$. La Table 1 montre que le couplage multicouches à sens direct permet d'obtenir de meilleurs résultats que le couplage proposé dans la littérature.

Ensemble	$h^{BP}_{uncoupled}$	$h^{MAP}_{uncoupled}$	$h^{BP}_{uniform}$	$h^{BP}_{forward}$
(3,6)	0.42944	0.48815	0.48808	0.48815
(4,8)	0.38345	0.49774	0.49442	0.49741
(5,10)	0.34155	0.49948	0.48268	0.49811
(6,12)	0.30746	0.49988	0.46031	0.49667

Table 1: Seuils BEC du couplage spatial multicouches à sens direct versus les seuils du couplage spatial comme proposé dans la littérature pour $w = 2$, $d_{bs} = 1$ et $d_{cs} = 2$.

Une autre illustration de la performance du couplage spatiale multicouches à sens direct est donnée Figure 4. Les seuils BEC sont tracés en fonction de la position spatiale i/L_c, pour $i = 1 \dots L_c$, pour l'ensemble $(3, 6; 1, 2; L_c)$. Les performances de l'ensemble couplé avec terminaison dépassent largement celles d'un ensemble LDPC $(3, 6)$ non couplé, mais aussi celles d'un ensemble $(4, 8)$ équivalent non couplé. En fait, l'ensemble couplé avec terminaison sature à 0.49741, i.e. très près du seuil MAP du code LDPC $(4, 8)$, se référer à la ligne du $(4, 8)$ dans la Table 1. Sans terminaison de la chaîne, le couplage multicouches à sens direct améliore le seuil, mais celui-ci reste bloqué à une valeur relativement faible de 0.44695.
Les résultats Figure 4 sont tracés pour 10000 itérations de décodage. Afin de rendre le seuil plat par rapport à la position spatiale i/L_c, le nombre d'itérations de décodage augmente avec L_c. De plus, le couplage multicouches à sens direct permet à un ensemble (d_b, d_c) en position i d'aider son voisin en position $i - 1$. La protection et la propagation du seuil se font donc de la droite vers la gauche en raison de la structure (d_{bs}, d_{cs}) du couplage spatial.

Chaîne couplée non uniformément et couplage spatialement variant

Dans cette section, des ensembles LDPC de paramètres différents (d_b et d_c) sont couplés. En effet, la structure non uniforme du couplage multicouches à sens direct nous permet de faire varier les ensembles LDPC à chaque position spatiale. Les ensembles LDPC locaux (d_b, d_c) restent réguliers (i.e d_b et d_c sont des entiers naturels), mais les ensembles ajoutés pour réaliser le couplage (d_{bs}, d_{cs}) ne sont pas nécessairement réguliers. L'objectif est de créer un escalier de seuils pour réaliser la protection inégale de l'information. En suivant les mêmes arguments et les mêmes notations que précédemment, nous construisons une chaîne non uniforme par collage d'ensembles LDPC différents.

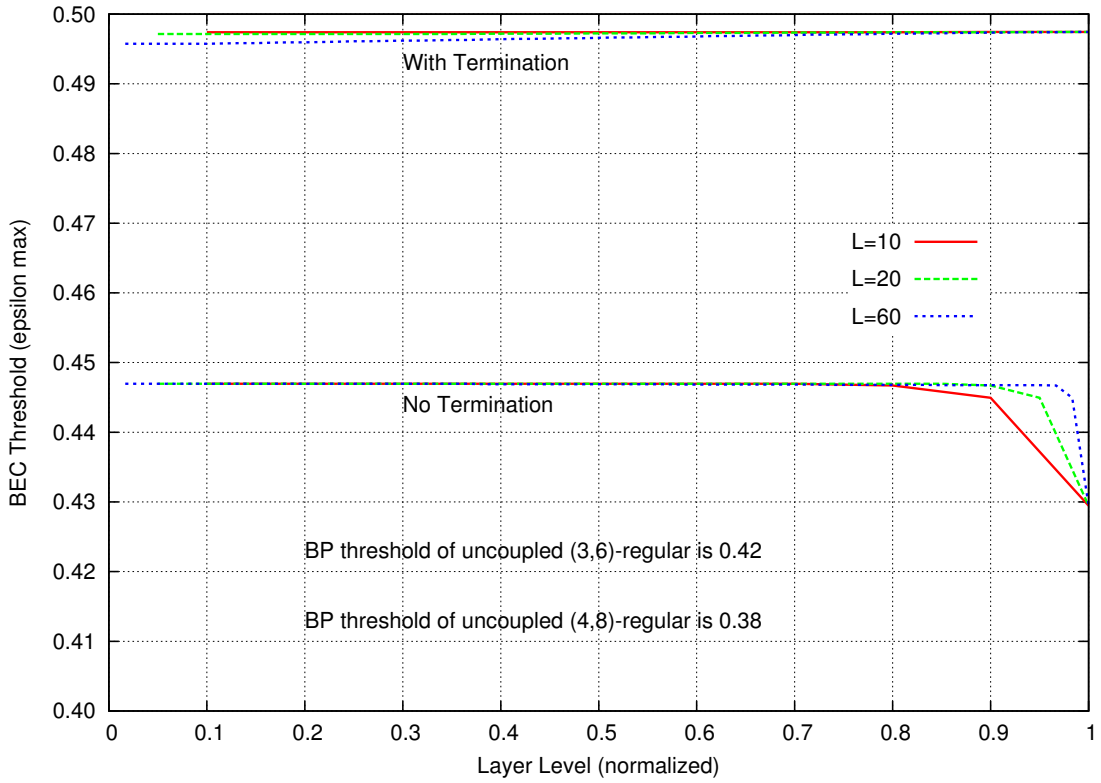


Figure 4: Performances d'un ensemble $(3, 6; 1, 2; L_c)$ avec un couplage multicouches à sens direct.

Definition Considérons deux ensembles couplés $(d_b, d_c; d_{bs}, d_{cs}; L_{c_1})$ et $(d'_b, d'_c; d'_{bs}, d'_{cs}; L_{c_2})$ correspondant à deux chaînes couplées de longueurs L_{c_1} et L_{c_2} respectivement. Le nouvel ensemble définit en collant les deux chaînes s'écrit comme

$$(d_b, d_c; d_{bs}, d_{cs}; L_{c_1}) \ \oplus \ (d'_b, d'_c; d'_{bs}, d'_{cs}; L_{c_2})$$

où la nouvelle chaîne a une longueur $L_c = L_{c_1} + L_{c_2}$.

En général, une chaîne de longueur $L_c = \aleph L_{c_1}$ est construite en collant $\aleph$ sous-chaînes. Le nouvel ensemble couplé peut s'écrire comme

$$\bigoplus_{j=1}^{\aleph} (d_b(j), d_c(j); d_{bs}(j), d_{cs}(j); L_{c_1}). \tag{2}$$

Considérons maintenant le plus grand nombre de sous-chaînes à l'intérieur d'une chaîne non uniforme. Cette situation extrême correspond à $\aleph = L_c$. La chaîne spatialement variante devient

$$\bigoplus_{i=1}^{L_c} (d_{b,i}, d_{c,i}; d_{bs,i}, d_{cs,i}; 1). \tag{3}$$

Les performances de ces ensembles couplés spatialement variant sont étudiées via la mise au point des équations à évolution de densité. Nous ne rentrerons pas dans les détails techniques de ces équations dans ce résumé.

Un escalier pour la protection inégale de l'information est tracé Figure 5. Les cinq ensembles LDPC non couplés sont, de gauche à droite, $(3, 12)$, $(3, 9)$, $(3, 6)$, $(3, 5)$ et $(3, 4)$. Les ensembles de couplage correspondant sont $(1, 4)$, $(1, 3)$, $(1, 2)$, $(1.2, 2)$ et $(1.5, 2)$. Le seuil en escalier sature sur le seuil de Shannon en escalier défini par les cinq seuils de Shannon des ensembles locaux.
Les performances du couplage spatialement variant sont tracées Figure 6 pour le canal à effacement binaire et $L_c = 200$.

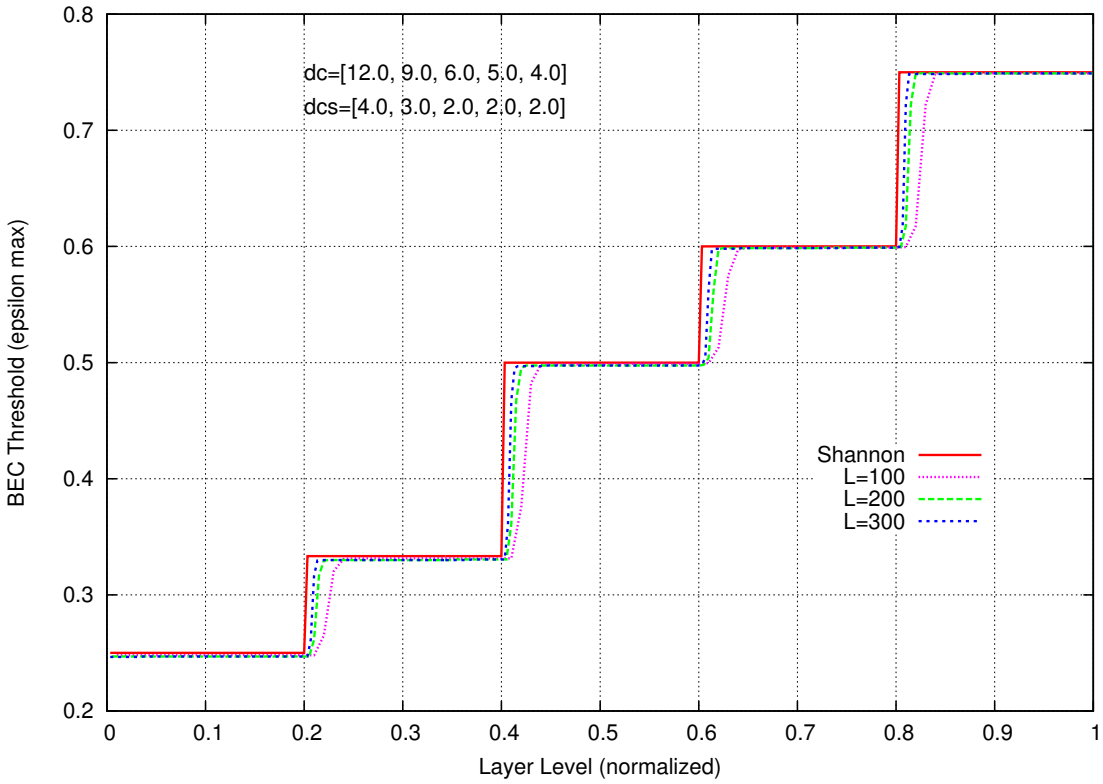


Figure 5: Seuils BEC pour une chaîne de couplage non uniforme multicouches à sens direct avec $\aleph = 5$ sous-chaînes chacune de longueur $L_{c_1} = L_c/5$.

Couplage spatial des ensembles Root-LDPC: Dopage des bits de parité

Nous allons maintenant appliquer le couplage spatial multicouches à sens direct aux codes Root-LDPC. Les codes Root-LDPC ont été conçus pour faire face aux évanouissements quasi-statiques [17][16]. L'idée principale est d'apporter plus de structure à un ensemble

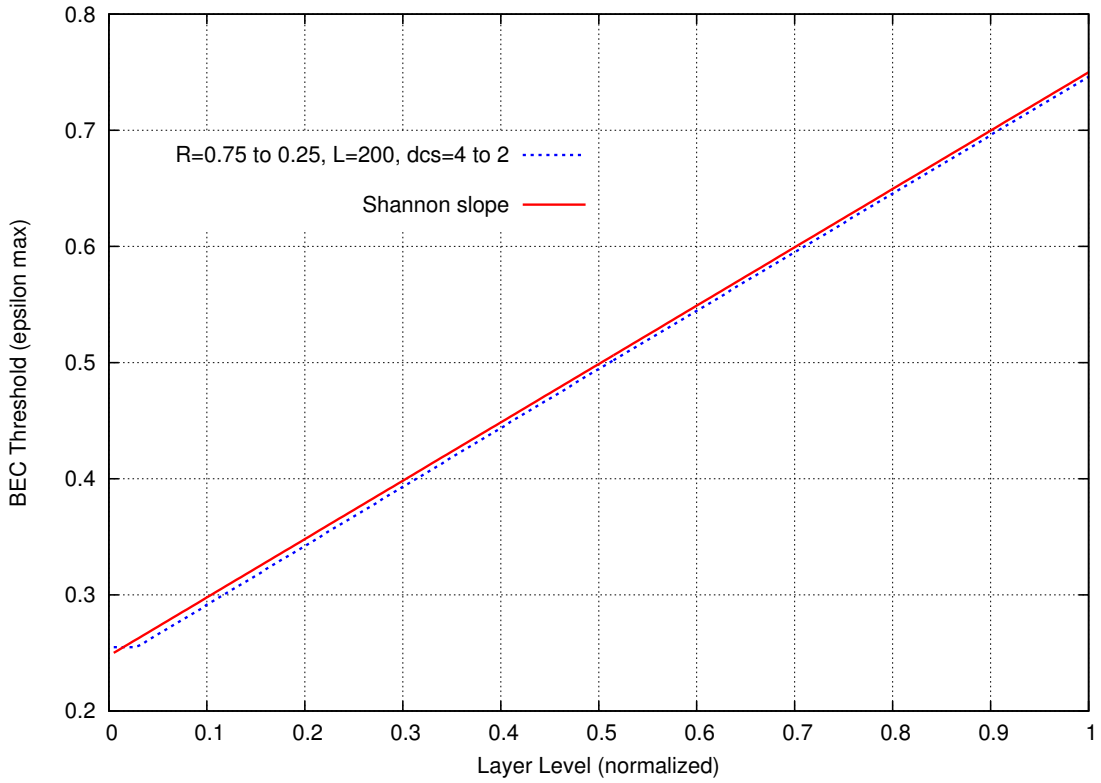


Figure 6: Seuils BEC pour une chaîne de couplage non uniforme multicouches à sens direct avec $\aleph = L_c$ ensembles locaux variant à chaque position spatiale. Les paramètres sont $L_c = 200$, $d_b = 3$, $d_c = 12$ et 4, $Rendement = 0.75$ à 0.25.

LDPC aléatoire afin de permettre aux nœuds d'information de recevoir les messages de tous les états du canal. Par conséquent, un ensemble Root-LDPC atteint une diversité qui est égale au nombre de degrés de liberté du canal à effacement par blocs. Définissions maintenant la notion de diversité.

Definition La diversité est rencontrée dans différents systèmes de communication selon deux aspects principaux:
• La diversité d'un point de vue des codes correcteurs d'erreurs. Elle consiste à créer plusieurs répliques de la même information.
• La diversité d'un point de vue du canal. Il correspond au nombre de degrés de liberté disponibles pour transmettre l'information.

Les codes Root-LDPC sont également MDS du point de vue de la borne du Singleton pour les évanouissements par blocs [16] et peuvent atteindre le plus grand rendement possible permettant d'obtenir une double diversité. Sous un décodage ML, une borne supérieure de l'ordre de diversité L est donnée par la borne du Singleton pour les

évanouissements par blocs:

$$L \leq 1 + \lfloor M(1-R) \rfloor, \tag{4}$$

où R est le rendement et M est la diversité intrinsèque du canal. Pour un rendement $R = 1/M$, les codes Root-LDPC peuvent atteindre une diversité pleine $L = M$.Le rendement maximal atteignable pour atteindre une diversité pleine est $R = 1/M$. Le rendement peut dépasser $1/M$ lorsque $L < M$.

Dans le cas du stockage distribué nous considérons une diversité $L = 2$ et un canal avec $M = 4$ évanouissements, nous pouvons donc concevoir des codes avec des rendements allant jusqu'à 3/4. Nous pouvons faire le parallèle entre les paramètres définis précédemment et ceux propres au stockage distribué. Nous considérons un centre de données avec un total de ℓ machines réparties sur $M = 4$ clusters. Un cluster représente donc un évanouissement du canal. Chaque cluster a sa propre probabilité d'effacement ϵ_i correspondant à un état parmi les quatre états d'effacements par blocs. Satisfaire une diversité L, dans le sens du stockage distribué, signifie que le code est capable de remplir tous les effacements si $L-1$ clusters sont effacés. Dans notre construction, les données stockées dans un cluster peuvent être complètement effacées, la diversité $L = 2$ garantit que tous ces effacements seront remplis. Le modèle du canal est représenté sur la Figure 7, où les N bits sont transmis sur les quatre BECs parallèles avec une probabilité d'effacement ϵ_i.

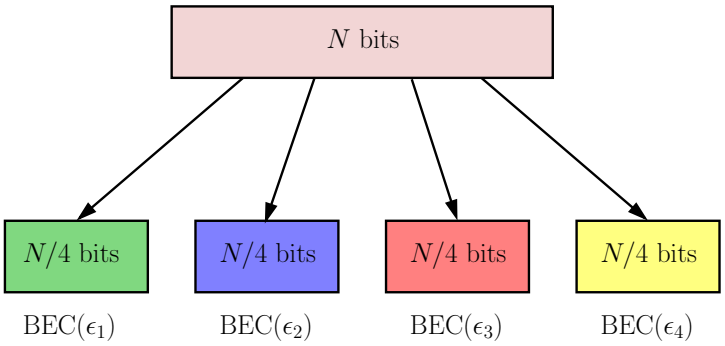


Figure 7: Canaux parallèles BECs où $N/4$ bits sont transmis sur chaque canal BEC(ϵ_i), $i = 1 \ldots 4$. N est la longueur du code.

Des fragments de $N/4$ bits du code Root-LDPC sont transmis sur chaque BEC (ϵ_i), $i = 1 \ldots 4$. Rappelons que pour les codes Root-LDPC, seuls les bits d'information sont connectés aux rootchecks (pour la définition d'un rootcheck voir [17]) et ont une diversité maximale. Cette propriété conduit à la reconstruction des données d'information en cas de défaillance d'un cluster. Cependant, le gain de codage d'un tel code est faible en raison d'un ordre de diversité inférieur pour les bits de parité. Afin d'améliorer le gain de codage et de saturer le seuil de coupure du code, nous proposons d'appliquer le couplage multicouches à sens direct aux ensembles Root-LDPC. Dans la suite, nous allons utiliser la terminologie de coloration pour décrire les quatre canaux BECs. Les éléments

binaires transmis sur un canal BEC(ϵ_i) seront appelés des nœuds variables possédant une couleur i, $i = 1 \dots 4$. Nous pouvons l'observer Figure 7, une couleur est associée à chaque BEC(ϵ_i).

Pour le modèle du canal décrit ci-dessus, nous divisons la première classe de bits (couleur verte) en bits d'information et en bits de parité, $1i$ et $1p$ respectivement. Une division similaire est faite pour les trois couleurs restant, menant à un total de huit classes de nœuds variables comme cela est illustré Figure 8 (sur la gauche du graphe de Tanner). Quatre classes de nœuds de contraintes sont établies sur la droite du graphe de Tanner puisque le canal possède $M = 4$ couleurs/évanouissements. Les nœuds contraintes $1c$, $2c$ et $3c$ sont des rootchecks pour les bits d'informations de type $1i$. En effet, nous pouvons constater Figure 8, que sur les nœuds contraintes $1c$, $2c$ et $3c$ uniquement les nœuds variables de type $1i$ appartiennent au premier évanouissement (les bits de parité $1p$ ne sont pas reliés à ces nœuds contraintes). On dit alors, que $1c$, $2c$ et $3c$ sont des rootchecks pour les bits d'informations de type $1i$. Les nœuds variables $1p$ sont connectés au non rootcheck $4c$. Le nombre d'arêtes allant des bits d'information et des bits de parité au nœud contrainte non rootcheck est Δ_i et Δ_p respectivement. La structure du code Root-LDPC est cyclique-symétrique. Les connexions des arêtes sont faites de manière identique pour les trois autres couleurs. L'ensemble illustré Figure 8 a un rendement 3/4 (rendement maximal atteignable pour garantir une diversité de deux).

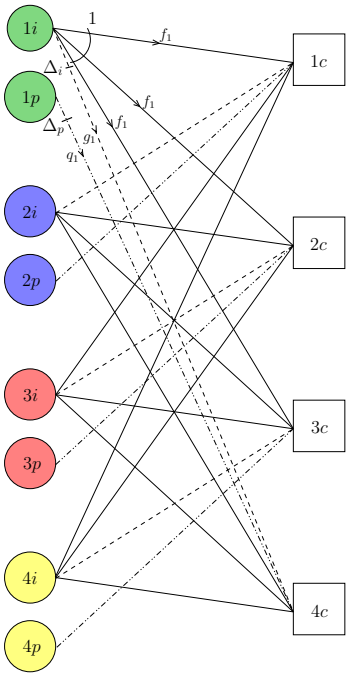


Figure 8: Représentation du graphe de Tanner d'un ensemble Root-LDPC de rendement 3/4 pour un canal avec quatre couleurs/évanouissements.

Ce code à double-diversité de rendement maximal avec quatre couleurs peut remplir des effacements indépendants. Cette capacité à manipuler à la fois les effacements indépendants et par blocs/couleurs est maintenant étudiée dans le plan (ϵ_1, ϵ_2). Afin d'étudier les performances de l'ensemble Root-LDPC dans le plan à effacement bidimensionnel, nous utilisons la propriété cyclique-symétrique du code en supposant que $\epsilon_2 = \epsilon_3 = \epsilon_4$. La frontière de coupure de la capacité est trouvée en écrivant que la capacité des quatre canaux BECs est égale au rendement du code, i.e. la frontière de coupure de la capacité est donnée par:

$$\epsilon_1 + 3\epsilon_2 = 1. \tag{5}$$

Les performances de l'ensemble Root-LDPC, dont le graphe de Tanner est représenté Figure 8, sont tracées Figure 9. On peut observer la double diversité en présence d'effacements par blocs sur la Figure 9. En effet, lorsque $\epsilon_1 = 1$ (la couleur 1 est complètement effacée) et $\epsilon_2 = 0$ (aucun effacement sur les trois autres sous-canaux), nous pouvons reconstruire intégralement les données d'information contenues sur le premier évanouissement.

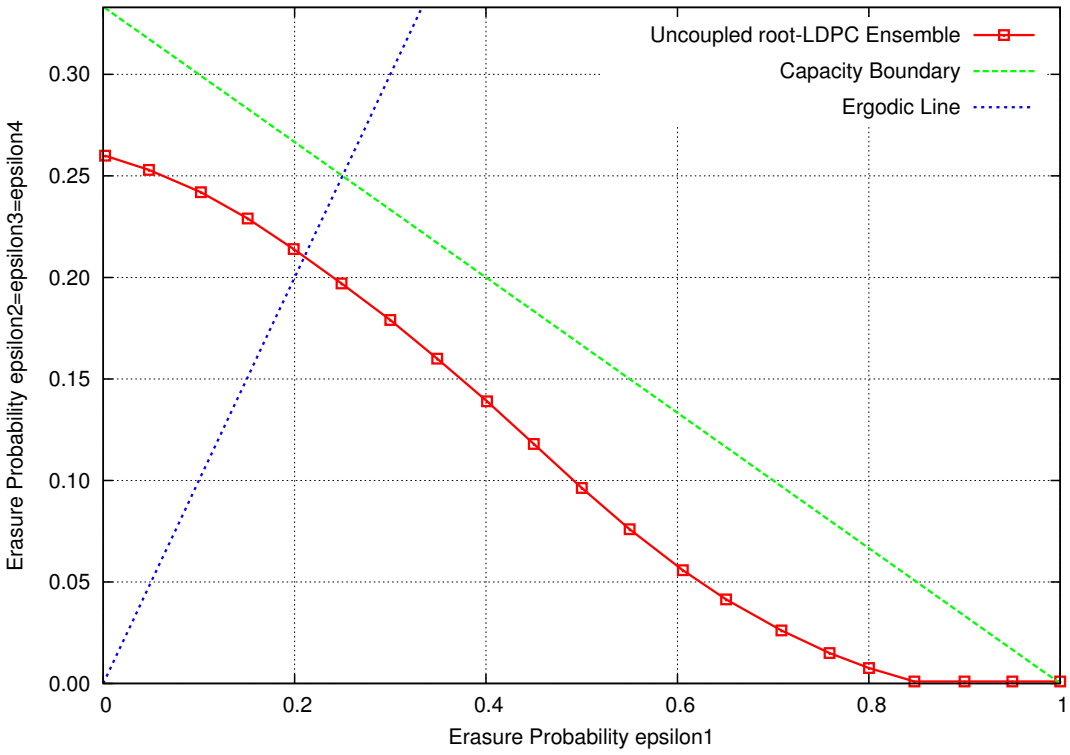


Figure 9: Frontière de coupure d'un code Root-LDPC de rendement 3/4 avec une double diversité. Les paramètres sont $\Delta_i = 2$ et $\Delta_p = 3$.

Nous proposons maintenant un couplage spatial partiel de cet ensemble Root-LDPC permettant d'améliorer la saturation du seuil à la frontière de coupure de la capacité. L'ensemble non couplé est copié L_c fois. En pratique, la longueur de la chaîne doit être assez grande, par exemple $L_c = 100$. Pour le schéma de couplage illustré Figure 10, la taille de la fenêtre de couplage est $w = 2$, chaque ensemble en position ℓ est couplé seulement à son voisin en position $\ell+1$. Dans le cas des codes Root-LDPC, seuls les bits de parité sont couplés au nœud contrainte non rootcheck en position spatiale suivante. En effet, la double diversité des bits d'information nous oblige à coupler uniquement les bits de parité. La chaîne est terminée à droite par un nœud contrainte supplémentaire. Dans le cas général, la taille de la fenêtre de couplage est w, les nœuds de parité en position ℓ sont couplés aux ensembles en position spatiale $\ell + 1, \ell + 2, ..., \ell + (w - 1)$. Dans ce cas, la chaîne est terminée en ajoutant $w-1$ nœuds contraintes qui correspondent à $w - 1$ positions spatiales supplémentaires à droite de la chaîne. Le nombre d'arêtes par nœud variable impliqué dans le couplage des bits de parité vers les $w - 1$ nœuds contraintes futures est Δ_c.

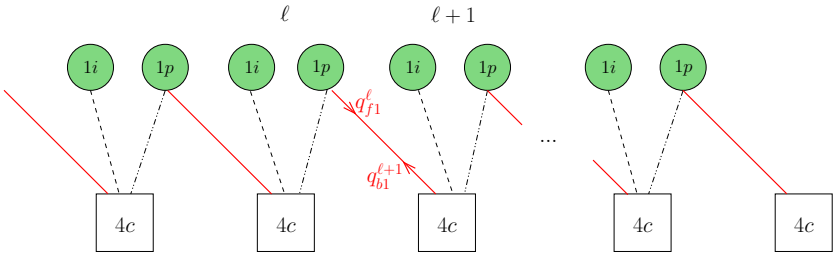


Figure 10: Structure de couplage spatial pour un ensemble Root-LDPC pour une fenêtre de couplage de taille $w = 2$. Seulement les bits de parité sont couplés au nœud contrainte en position spatiale suivante.

La frontière de coupure pour un ensemble Root-LDPC couplé est représentée Figure 11 pour une fenêtre de couplage de taille $w = 2$,3 et 5. Néanmoins, nous n'observons pas une saturation assez forte des codes Root-LDPC couplés. Ceci est dû au couplage spatial partiel des bits de parité uniquement.

Conclusions

Nous avons proposé dans ce chapitre une nouvelle méthode de couplage spatial, le couplage multicouches à sens direct. Cette technique donne des résultats impressionnants avec une taille minimale de la fenêtre de couplage et permet de réaliser un couplage spatialement variant. Le couplage spatialement variant est réalisé en collant des protographes LDPC de paramètres différents, ce qui conduit à un seuil BP spatialement variable. Nous avons ensuite appliqué cette technique aux ensembles Root-LDPC afin de satisfaire aux exigences et contraintes du stockage distribué.

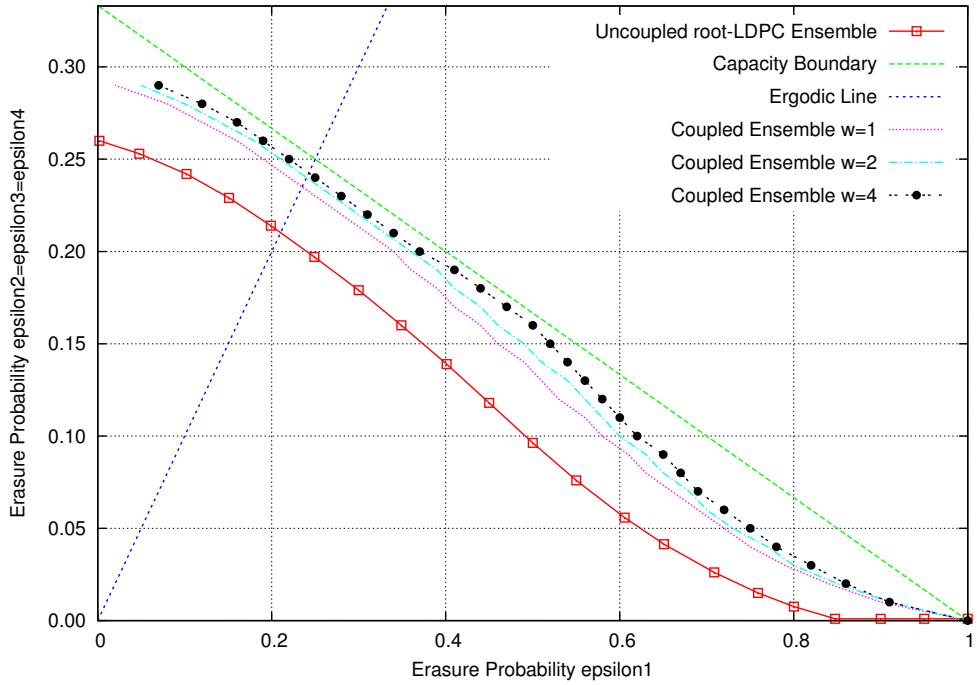


Figure 11: Performances d'un ensemble couplé Root-LDPC de rendement 3/4 avec des fenêtres de couplage de taille $w = 2, 3, 5$. Les paramètres du graphe sont $\Delta_c = w - 1$, $\Delta_i = 2$ et $\Delta_p = 3, 5, 10$.

Afin de conclure sur cette partie, nous pouvons comparer la localité d'un ensemble Root-LDPC à celui d'un ensemble LDPC aléatoire dans le cas d'effacements indépendants et d'effacements par blocs. Les résultats sont résumés dans le Tableau ci-dessous:

Localité	LDPC aléatoire	Root-LDPC
Effacements indépendants	$d_c - 1$	$d_c - 1$
Effacements par blocs	$O(\log n)$	$d_c - 1$

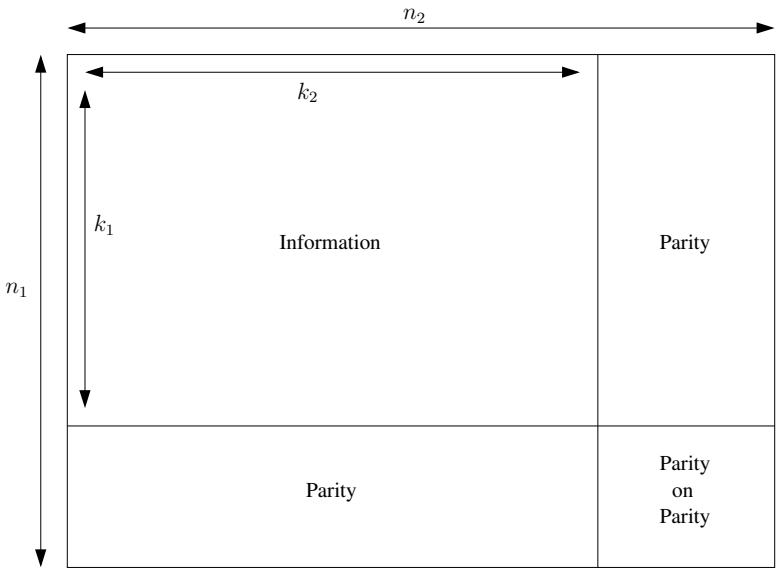
Les codes Root-LDPC, grâce à la structure de rootchecks, sont plus adaptés au stockage distribué dans le cas d'effacements par blocs (couleurs). En effet, la localité est beaucoup plus faible pour les codes Root-LDPC, dans le cas d'effacements par blocs, que pour les codes LDPC aléatoires.

Dans ce chapitre, nous avons étudié les ensembles LDPC spatialement couplés et les ensembles Root-LDPC spatialement couplés. Ces codes permettent d'atteindre un bon compromis entre une bonne localité et un décodage itératif. Cependant, les codes LDPC et Root-LDPC ne sont pas adaptés pour les applications de stockage distribué nécessitant une mise à jour des données. En effet, en termes de complexité la mise à jour des données est équivalente à refaire un codage total des données. Or, le processus

de codage est lourd car la matrice de parité des codes LDPC étant creuse leur matrice génératrice est dense. Dans le chapitre suivant, un nouveau schéma de codage utilisant les codes produits à double diversité est proposé afin de remédier à ce problème.

Chapitre 3: Coloration d'arêtes sur le graphe d'un code produit

Dans ce chapitre, nous étudierons les codes produits. Un code produit bi-dimensionel est le produit de Kronecker de deux codes élémentaires linéaires en blocs C_1 et C_2: $C_P = C_1[n_1, k_1]_q \otimes C_2[n_2, k_2]_q$, où C_1 est le code colonne et C_2 est le code ligne. Voici la représentation matricielle d'un code produit:



Le code produit a une longueur $N = n_1 n_2$ et une dimension $K = k_1 k_2$. Nous nous limiterons à l'étude des codes produits réguliers.

Représentation d'un code produit

Un code produit peut être représenté par un graphe bipartite complet non compact (V_1, V_2, E) possédant n_1 sommets à gauche, n_2 sommets à droite, et $N = |E| = n_1 n_2$ arêtes représentant les symboles du code. Le graphe étant bipartite complet, les sommets à gauche ont un degré n_2 et à droite n_1. Les dimensions k_1 et k_2 des codes élémentaires n'ont pas d'effet sur le nombre de sommets et d'arêtes du graphe du code produit. La Figure 12 est un exemple de graphe bipartite d'un code produit régulier symétrique $[4, 2] \otimes [4, 2]$.

Boutros et al. ont introduit la représentation en graphe compact d'un code produit dans [14]. Cette représentation compacte a beaucoup d'avantages, le principal étant

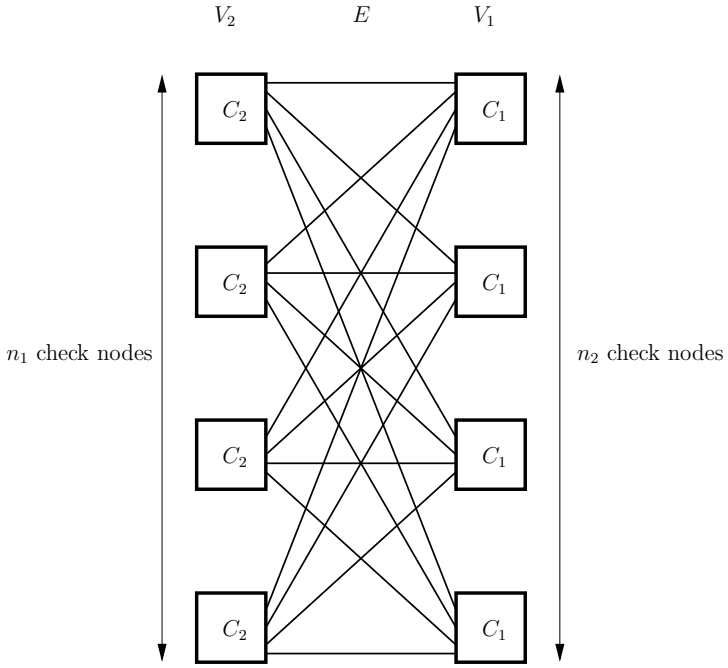


Figure 12: Graphe non compact bipartite $\mathcal{G} = (V_1, V_2, E)$ d'un code produit $[4, 2]^{\otimes 2}$, i.e. $n_1 = n_2 = 4$, $k_1 = k_2 = 2$, $|V_1| = |V_2| = 4$ et $|E| = N = n_1 n_2 = 16$ arêtes représentant 16 symboles dans $\mathbb{F}_q$.

sa capacité à imiter un graphe de Tanner avec les nœuds de contrôle de parité. Cette représentation a été introduite pour la diversité. Les sommets (réciproquement arêtes) du graphe non compact sont groupés en super-sommets (super-arêtes). Pour créer un super-sommets $n - k$ sommets sont rassemblés. La Figure 13 représente le graphe compact d'un code produit $[4, 2]^{\otimes 2}$. Le nombre d'arêtes dans le graphe compact est donné par:

$$N^c = |E^c| = \left\lceil \frac{n_1}{n_1 - k_1} \right\rceil \times \left\lceil \frac{n_2}{n_2 - k_2} \right\rceil . \tag{6}$$

Notre but est maintenant d'effectuer une coloration d'arêtes à double diversité sur le graphe compact d'un code produit. Notons $\Phi(E)$ l'ensemble des colorations sur le graphe non compact et $\Phi(E^c)$ l'ensemble des colorations sur le graphe compact. Considérons une palette de M couleurs, le nombre total de colorations possibles sur le graphe non compact est donné par

$$|\Phi(E)| = \frac{N!}{((N/M)!)^M}, \tag{7}$$

et sur le graphe compact

$$|\Phi(E^c)| = \frac{N^c!}{((N^c/M)!)^M}. \tag{8}$$

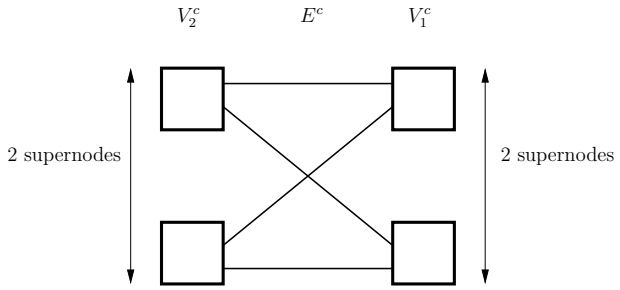


Figure 13: Graphe compact bipartite $\mathcal{G}^c = (V_1^c, V_2^c, E^c)$ avec deux supernœuds de chaque côté pour les codes produits de la forme $[n, n/2]^{\otimes 2}$, par exemple $|V_1^c| = |V_2^c| = 2$ et $|E^c| = N^c = 4$ super-symboles. Chaque super-symbole (i.e. super-arête) contient $n^2/4$ symboles.

Prenons l'exemple du code produit $[12, 10]^{\otimes 2}$ avec $M = 4$ couleurs, il y a $2 \cdot 10^{83}$ possibilités de colorations sur le graphe non compact et $2 \cdot 10^{19}$ sur le graphe compact. Il est donc clair que la construction de codes produits pour la diversité est beaucoup plus facile quand elle est basée sur le graphe compact $\mathcal{G}^c = (V_1^c, V_2^c, E^c)$, le nombre de colorations possibles du graphe étant réduit.

Propriétés et décodage d'un code produit

D'une manière similaire aux codes Root-LDPC, nous introduisons la notion de symboles racines (root symboles, similaire à la notion de rootchecks) pour les codes produits.

Definition Soit $\mathcal{G}^c$ le graphe compact du code produit, ϕ une coloration donnée, et $e \in E^c$ un super-symbole. e est un *root super-symbole* (super-symbole racine) avec le respect de $\phi(e)$, si il admet un sommet voisin v, $v \in V_1^c$ ou $v \in V_2^c$, tel que toutes les arêtes adjacentes f dans v satisfassent $\phi(f) \neq \phi(e)$.

Tous les symboles d'un code produit ne sont pas des symboles racines. En effet, sous un décodage itératif ligne-colonne, sur un canal à effacement par blocs, certains symboles peuvent être résolus en deux itérations de décodage ou plus. Certains ensembles de symboles ne peuvent jamais être résolus et sont appelés des ensembles d'arrêt [27][93][88]. Notre étude est limitée à une double diversité, $L = 2$, i.e. à effacer tous les symboles d'une couleur parmi les M. Une absence de diversité équivaut à $L = 1$.
Nous établissons maintenant la définition de l'ordre racine (root-order) ρ d'un symbole. Pour les symboles racines, satisfaisant la définition précédente, l'ordre racine est $\rho = 1$. L'ordre racine d'un symbole pouvant être résolu après deux itérations de décodage est $\rho = 2$. L'ordre racine peut être défini très simplement comme le nombre minimum d'itérations de décodage nécessaire à la résolution d'un symbole. La définition de l'ordre racine ρ peut aussi être formulée de la manière récursive suivante (pour $\rho \geq 2$).

Definition Soit $\mathcal{G}^c$ le graphe compact d'un code produit, $\phi \in \Phi(E^c)$ une coloration d'arêtes donnée, et $e \in E^c$ un super-symbole. e a *un ordre racine* $\rho(e) = \min(\rho_1, \rho_2)$ où:
1- Soit $v_1 \in V_1^c$ le sommet voisin colonne de e. $\forall f$ adjacent à e dans v_1 et $\phi(f) = \phi(e)$, nous avons $\rho(f) < \rho_1$.
2- Soit $v_2 \in V_2^c$ le sommet voisin ligne de e. $\forall f$ adjacent à e dans v_2 et $\phi(f) = \phi(e)$, nous avons $\rho(f) < \rho_2$.

La définition précédente implique que $\rho(e) = 1$ si il n'existe pas d'arête adjacente de la même couleur arrivant au même sommet. De plus, pour une arête e n'admettant pas un ordre fini $\rho(e)$, nous fixons $\rho(e) = \infty$. Dans le cas où $\rho(e) = \infty$, ce symbole ne pourra jamais être décodé même après un nombre infini d'itérations de décodage. On dit que ce symbole appartient à un ensemble d'arrêt.

Nous pouvons maintenant définir l'ordre racine maximal d'un code produit. Le paramètre ρ_{max} est important en pratique afin de donner une borne supérieure de la quantité d'information transmise dans un réseau.

Corollaire *Soit C_P un code produit $[n_1, k_1] \otimes [n_2, k_2]$ avec un graphe compact $\mathcal{G}^c$. Soit $\phi \in \Phi(E^c)$ une coloration d'arêtes. Nous définissions*

$$\rho_{max}(\phi) = \max_{e \in E^c} \rho(e). \tag{9}$$

C_P atteint une double diversité sous un décodage itératif ligne-colonne si et seulement si $\rho_{max}(\phi) < \infty$. Dans ce cas, nous disons que ϕ est une coloration à double diversité et tout symbole $e \in E^c$ peut être résolu après au plus ρ_{max} itérations de décodage avec $\rho_{max}(\phi) \leq \rho_u$.

Nous pouvons maintenant définir la localité d'un code produit. On rappelle que la localité est le nombre de symboles à contacter afin de réparer un symbole effacé. En vertu du décodage algébrique ligne-colonne des codes élémentaires, la localité d'un code produit par itération de décodage est $\max(n_1, n_2)$. Si l'on considère un décodage ML des codes élémentaires MDS, la localité devient $\max(k_1, k_2)$. Finalement pour un code de produit avec décodage algébrique ligne-colonne, le transfert d'information par symbole est borné par

$$\rho_{max}(\phi) \times \max(n_1, n_2). \tag{10}$$

Nous allons maintenant introduire une définition simple et générale d'un ensemble d'arrêt pour un code linéaire.

Definition Soit $C[n, k]_q$ un code linéaire. Supposons que les symboles du mot de code soient transmis sur un canal à effacement et que le décodeur $\mathcal{D}$ utilise une méthode de décodage déterministe. Considérons un ensemble $\mathcal{S}$ de s positions fixées $i_1, i_2, \ldots, i_s$ où $1 \leq i_j \leq n$. L'ensemble $\mathcal{S}$ est dit être un ensemble d'arrêt si $\mathcal{D}$ ne peut pas retrouver le mot de code transmis lorsque tous les symboles sur les s positions données par $\mathcal{S}$ sont

effacés.

La définition exacte d'un ensemble d'arrêt dépend du décodage itératif. Pour les codes produits, quatre méthodes de décodage sont connues:

- Type I: Décodeur ML. Il s'agit d'un décodeur non itératif. Il est basé sur une réduction de Gauss de la matrice de contrôle de parité du code produit.
- Type II: Décodeur algébrique itératif. Lors des itérations de décodage impaires, les codes élémentaires C_1 de chaque colonne sont décodés par un décodeur algébrique qui remplit jusqu'à $d-1$ effacements. De même, lors des itérations paires, les codes élémentaires C_2 sur chaque ligne sont décodés par un décodeur algébrique.
- Type III: Décodeur itératif ML-par-code-élémentaire. Ce décodeur a été étudié par Rosnes dans [88] pour les codes produits binaires. Lors des itérations de décodage impaires, les codes colonnes C_1 sont décodés par un décodeur optimal (ML pour C_1). Lors des itérations de décodage paires, les codes lignes C_2 sont décodés par un décodeur optimal similaire (ML pour C_2).
- Type IV: Décodeur itératif à propagation de croyance basé sur le graphe de Tanner de $C_{\mathcal{P}}$, comme étudié par Schwartz et al. pour les codes linéaires en blocs [93] et Di et al. pour les codes LDPC [27].

Les trois décodeurs itératifs énumérés ci-dessus donnent lieu à trois types différents d'ensemble d'arrêt. Il est prouvé dans cette thèse par le Corollaire 4.2 et la Prosposition 4.2, que les ensembles d'arrêt de type II et de type III sont identiques si les codes élémentaires sont MDS.

Définition, propriétés et énumération des ensembles d'arrêt

Cette section est consacrée à la définition des notions primordiales pour comprendre et interpréter ce qu'est un ensemble d'arrêt. Nous commençons par définir le support d'un code linéaire. Cette notion de support $\mathcal{X}$ est appliquée aux codes élémentaires lignes et colonnes du code produit.

Définition Soit C un q-aire code linéaire de longueur n, i.e. C est un sous-espace de $\mathbb{F}_q^n$ de dimension k. Le support de C, noté $\mathcal{X}(C)$, est l'ensemble des ℓ positions distinctes $\{i_1, i_2, \ldots, i_\ell\} = \{i_j\}_{j=1}^{\ell}$, $1 \leq i_j \leq n$, tel que pour tout j, il existe un mot de code $c = (c_1 \ldots c_n) \in C$ avec $c_{i_j} \neq 0$.

Définissons maintenant le support rectangulaire qui est utilisé pour représenter un ensemble d'arrêt dans un code produit à deux dimensions.

Définition Soit $\mathcal{S} \subseteq \{1, \ldots, n_1\} \times \{1, \ldots, n_2\}$ un ensemble de positions des symboles dans le code produit. L'ensemble des positions des lignes associé à $\mathcal{S}$ est $\mathcal{R}_1(\mathcal{S}) = \{i_1, \ldots, i_{\ell_1}\}$ où $|\mathcal{R}_1(\mathcal{S})| = \ell_1$ et pour tout $i \in \mathcal{R}_1(\mathcal{S})$ il existe $(i, \ell) \in \mathcal{S}$. L'ensemble des

positions des colonnes associé à $\mathcal{S}$ est $\mathcal{R}_2(\mathcal{S}) = \{j_1, \ldots, j_{\ell_2}\}$ où $|\mathcal{R}_2(\mathcal{S})| = \ell_2$ et pour tout $j \in \mathcal{R}_2(\mathcal{S})$ il existe $(\ell, j) \in \mathcal{S}$. Le support rectangulaire de $\mathcal{S}$ est

$$\mathcal{R}(\mathcal{S}) = \mathcal{R}_1(\mathcal{S}) \times \mathcal{R}_2(\mathcal{S}), \tag{11}$$

i.e. le support rectangulaire est le plus petit rectangle $\ell_1 \times \ell_2$ contenant toutes les lignes et colonnes de $\mathcal{S}$.

Une définition formelle d'un ensemble d'arrêt est donné ci-dessous

Définition Considérons un code produit $C_P = C_1 \otimes C_2$. Soit $\mathcal{S} \subseteq \{1, \ldots, n_1\} \times \{1, \ldots, n_2\}$ avec $|\mathcal{R}_1(\mathcal{S})| = \ell_1$ and $|\mathcal{R}_2(\mathcal{S})| = \ell_2$. Considérons les ℓ_1 lignes de $\mathcal{S}$ données par $\mathcal{S}_r^{(i)} = \{j : (i,j) \in \mathcal{S}\}$ et les ℓ_2 colonnes de $\mathcal{S}$ données par $\mathcal{S}_c^{(j)} = \{i : (i,j) \in \mathcal{S}\}$. L'ensemble $\mathcal{S}$ est un ensemble d'arrêt de type III pour C_P si il existe des sous codes linéaires $C_c^{(j)} \subseteq C_1$ et $C_r^{(i)} \subseteq C_2$ tel que $\mathcal{X}(C_c^{(j)}) = \mathcal{S}_c^{(j)}$ et $\mathcal{X}(C_r^{(i)}) = \mathcal{S}_r^{(i)}$ pour tout $i \in \mathcal{R}_1(\mathcal{S})$ et pour tout $j \in \mathcal{R}_2(\mathcal{S})$.

La cardinalité de $|\mathcal{S}|$ est appelée la taille de l'ensemble d'arrêt et sera aussi référencée dans la suite comme le poids de l'ensemble d'arrêt.

Les ensembles d'arrêt pour les décodeurs de types II-IV peuvent être caractérisés par quatre propriétés principales:

- Les ensembles d'arrêt évidents (obvious) ou non évidents (not-obvious), aussi connus comme ensembles de rang 1. Un ensemble d'arrêt $\mathcal{S}$ est évident si $\mathcal{S} = \mathcal{R}(\mathcal{S})$.
- Les ensembles d'arrêt primitifs ou non primitif. Un ensemble d'arrêt est primitif si il ne peut pas être partitionné en deux ou plusieurs ensembles d'arrêt plus petits.
- Les ensembles d'arrêt mot de code ou non mot de code. Un ensemble d'arrêt $\mathcal{S}$ est dit être un mot de code si il existe un mot de code c dans C_P tel que $\mathcal{X}(c) = \mathcal{S}$.
- Les ensembles d'arrêt ML-corrigeable ou non ML-corrigeable. Un ensemble d'arrêt $\mathcal{S}$ ne peut pas être corrigé via un décodeur ML si il inclut le support d'un mot de code non nul.

Nous proposons dans ce travail de de thèse une énumération et caractérisation complète des ensembles d'arrêt pour les codes produits à codes élémentaires MDS. Cette énumération est faite jusqu'à un poids $(d_1 + 1)(d_2 + 1)$, où d_1, d_2 sont les distances de Hamming des codes élémentaires. Cette énumération des ensembles d'arrêt va au-delà du poids $d_1 d_2 + \max(d_1, d_2)$ du théorème 3 de Tolhuizen dans le cas de l'énumération des mots de codes [105]. Nous passerons les détails de cette énumération dans ce résumé, tout est expliqué en détail Chapitre 4, Lemmes 4.3&4.4 et Théorèmes 4.2&4.3.

Algorithme à évolution différentielle de colorations d'arêtes

Nous proposons dans cette section un algorithme pour les codes produits permettant de chercher une coloration d'arêtes avec un grand nombre d'arêtes d'ordre 1 (*good edges*) et atteignant une double diversité. La recherche est faite sur l'ensemble des colorations d'arêtes $\Phi(E^c)$ sur le graphe compact $\mathcal{G}^c$. La boucle principale de notre algorithme est une boucle à évolution différentielle qui fait muter/évoluer une fraction de la population des mauvaises arêtes. Cet algorithme est appelé algorithme à évolution différentielle de colorations d'arêtes (DECA).

La population des mauvaises arêtes est définie par l'ensemble suivant

$$B = \{e \in E^c : \rho(e) > 1\}. \tag{12}$$

Le nombre de bonnes arêtes est donné par

$$\eta(\phi) = |E^c \setminus B| = |\{e \in E^c : \rho(e) = 1\}|. \tag{13}$$

Parmi les $|B|$ mauvaises arêtes, la couleur d'une fraction $\aleph$ des arêtes est modifiée dans le but de maximiser $\eta(\phi)$, $\aleph \in \mathbb{N}$. La fraction $\aleph/|B|$ doit être suffisamment large pour permettre une évolution de la population mais elle doit rester suffisamment petite dans le but de limiter la complexité de l'algorithme. Le fonctionnement de l'algorithme DECA est décrit par l'organigramme Figure 14.

La complexité de DECA est majoritairement due à la boucle à évolution différentielle. La complexité est proportionnelle au nombre total de permutations $\Lambda(\gamma_1, \ldots, \gamma_M)$ par itération associées à chaque composition faible $\Gamma : \gamma_1 + \ldots + \gamma_M = \aleph$. Par conséquent, le nombre d'opérations dans DECA se comporte comme

$$\Lambda \leq \Lambda_{max}(\aleph, M) = \frac{\aleph!}{((\aleph/M)!)^M}. \tag{14}$$

Lorsque $\aleph$ n'est pas un multiple de M, le dénominateur dans le terme de droite doit être re-écrit comme $\prod_{i=1}^{i_0} \lfloor \aleph/M \rfloor \times \prod_{i=i_0+1}^{M} \lceil \aleph/M \rceil$, où i_0 est choisi tel que la somme de tous les éléments impliqués dans les deux produits soit égale à $\aleph$. Toutes les compositions Γ de $\aleph$ ne sont pas considérées dans cet algorithme. En fait, le nombre total de permutations pour toutes les compositions faibles est

$$\sum_{j=1}^{\Gamma} \frac{(\gamma_1(j) + \ldots + \gamma_M(j))!}{\prod_{i=1}^{M} \gamma_i(j)!} = M^{\aleph}. \tag{15}$$

Heureusement, la complexité par itération de DECA donnée dans (14) est bien plus petite que $M^{\aleph}$, i.e. $\Lambda_{max} = o(M^{\aleph})$. Dans la conception pratique des codes produits, nous avons $\Lambda_{max} \ll M^{\aleph} \ll M^{N^c}$.
Le but de l'algorithme proposé est de maximiser $\eta(\phi)$ mais il ne peut pas garantir que $\forall e \in E^c, \rho(e) < \infty$. Dans certains cas, l'algorithme termine toutes ses itérations et il

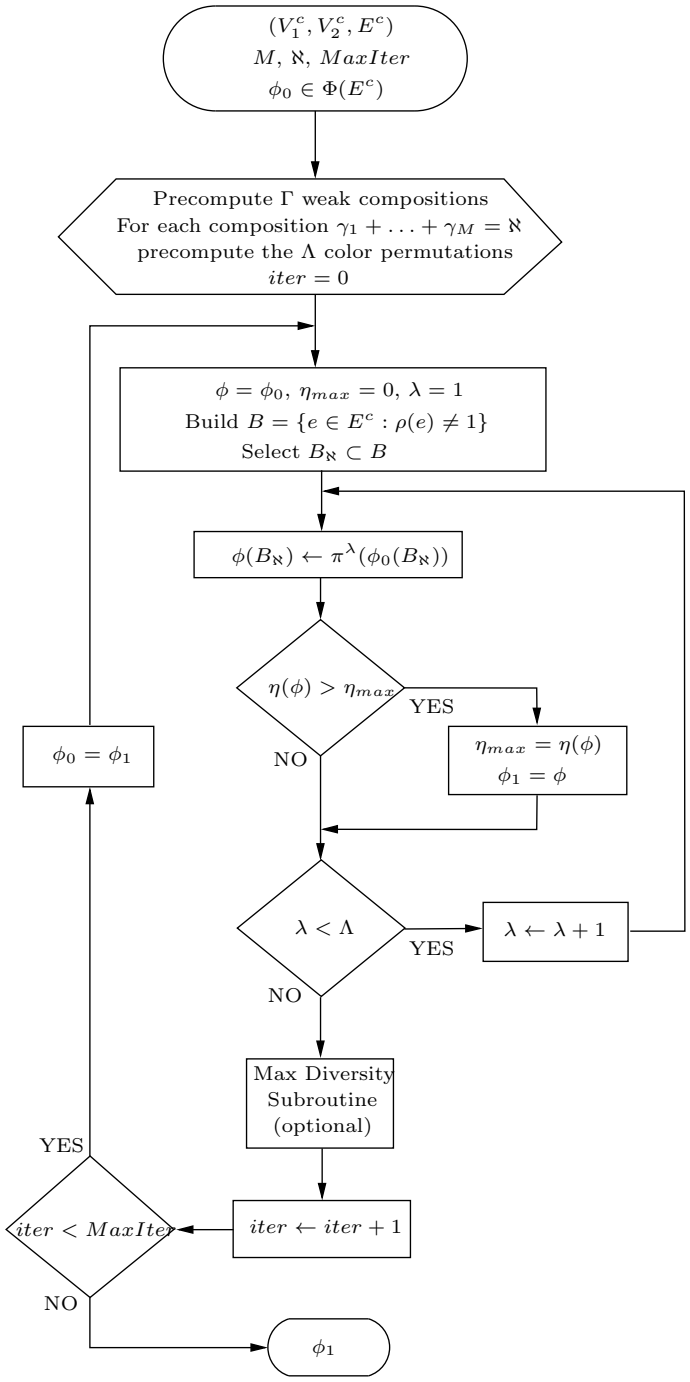


Figure 14: Organigramme de l'algorithme DECA pour concevoir un code produit à double diversité.

reste encore quelques arêtes d'ordre infini, i.e. le coloriage n'est pas à double diversité. Ceci apparaît lorsque l'on essaie de concevoir un code produit de rendement très proche ou égal à $1 - 1/M$. Pour remédier à cette faiblesse, DECA possède une sous-routine appelée *Max Diversity*, voir Figure 14. De même que pour la seconde étape de la boucle à évolution différentielle, cette sous-routine applique une permutation de couleurs sur les arêtes du sous-ensemble $B_{\aleph_1}$, $|B_{\aleph_1}| = \aleph_1$, $B_{\aleph_1} \subset B^\infty$, et

$$B^\infty = \{e \in E^c : \rho(e) = \infty\}. \tag{16}$$

Appliquons maintenant DECA pour concevoir un code produit à codes élémentaires MDS ayant une double diversité. Les valeurs numériques ont été sélectionnées de façon à rendre ces codes appropriés pour les applications de stockage distribué et les systèmes à diversité dans les réseaux sans fil. Le paramètre $MaxIter$ est fixé à 100. DECA avec une centaine d'itérations trouve une solution en une petite fraction de seconde sur un ordinateur standard.

Exemple DECA est appliqué à la coloration d'arêtes du graphe compact d'un code produit $C_{P1} = [n, k, d]_q^{\otimes 2}$, où $n = 12$, $k = 10$, $d = 3$, et la taille de l'alphabet du corps fini est $q > 12$. Le rendement de C_{P1} est $R(C_{P1}) = 25/36 < 1 - 1/M = 3/4$, i.e. l'écart à (4) est de $1/18$. Ce petit écart est suffisant pour rendre une conception à double diversité simple. La coloration dans $\Phi(E^c)$ peut facilement être convertie en son homologue dans $\Phi(E)$ en remplaçant chaque super-symbole par quatre symboles.
De (7) et (8) nous déduisons que le nombre total de colorations d'arêtes est $|\Phi(E)| \approx 10^{83}$ sur le graphe non compact et $|\Phi(E^c)| \approx 10^{19}$ sur le graph compact. Le paramètre à évolution différentielle $\aleph$ est fixé à 8. La sous-routine à diversité est désactivée. Nous avons

$$\Lambda_{max}(8, 4) = 2520 \ll |\Phi(E^c)| \ll |\Phi(E)|.$$

Pour presque tous les choix de colorations initiales ϕ_0 uniformément distribués dans $\Phi(E^c)$, DECA atteint une coloration ϕ_1 à double diversité. Pour à peu près un choix parmi trois de ϕ_0, l'algorithme retourne une coloration ϕ_1 telle que $\eta(\phi_1) \geq 28$. La Figure 15(a) montre la représentation matricielle d'une coloration ϕ_1 trouvée par DECA. Elle possède 32 super-symboles d'ordre 1 ce qui correspond à $\eta = 128$ symboles d'ordre 1 dans (V_1, V_2, E). Les ordres racines associés sont affichés sur la Figure 15(b). L'ordre le plus haut atteint pour cette coloration est $\rho_{max}(\phi) = 2$. Cela signifie que chaque super-symbole est décodé en au plus deux itérations de décodage.

L'efficacité de l'algorithme DECA a été validée dans l'exemple précédent en termes de nombre d'arêtes d'ordre 1 et de l'ordre racine maximal atteint parmi toutes les arêtes. De toute évidence, tout en évoluant d'une coloration à une autre afin d'obtenir le plus grand $\eta(\phi)$ possible, DECA a également produit un très petit ordre maximal $\rho_{max}(\phi)$. Il est aussi important de noter que toutes les constructions déterministes semblent être vouées à l'échec étant donné la taille considérable des ensembles $\Phi(E)$ et $\Phi(E^c)$.

G	B	B	B	Y	R
G	B	R	G	Y	G
B	G	G	Y	B	R
G	Y	Y	R	R	B
Y	B	R	Y	G	Y
R	R	R	Y	G	B

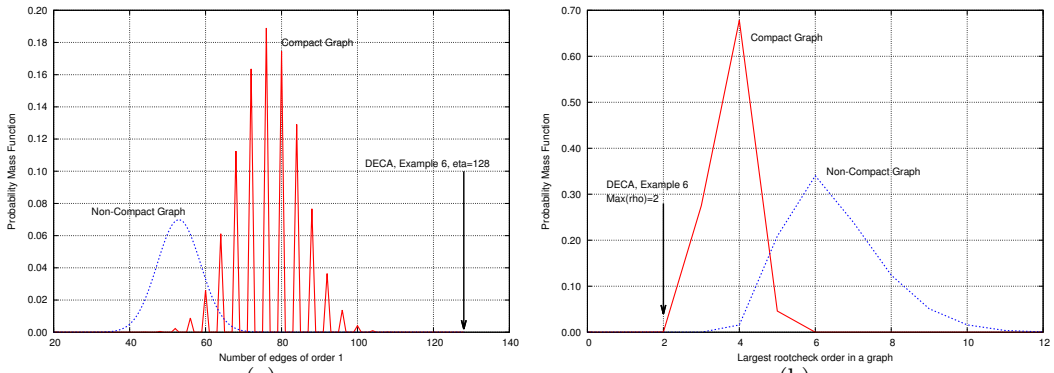
(a)

1r	2b	1c	1c	1r	1r
2b	1r	1r	1c	1r	1c
1c	1c	1c	1r	1c	1r
1r	1c	1c	1c	1c	1r
1c	1r	1r	2b	1r	1c
1c	1c	2b	1r	1r	1r

(b)

Figure 15: Matrice compacte de coloration (figure a) et les ordres racines associés (figure b) pour le code produit $C_{P1} = [12, 10]^{\otimes 2}$ trouvés par DECA, $\eta(\phi) = 32$ et $\rho_{max} = 2$.

Un autre moyen de démontrer l'efficacité de l'algorithme DECA est de comparer le résultat obtenu à une sélection aléatoire de $\Phi(E)$ et $\Phi(E^c)$ et d'obtenir une estimation des distributions de $\eta(\phi)$ et $\rho_{max}(\phi)$. En effet, une permutation uniformément répartie dans le groupe symétrique d'ordre N donne une coloration des arêtes uniformément distribuée. Ceci est également vrai pour $\Phi(E^c)$ lorsque le groupe symétrique est d'ordre N^c. Ainsi, nous avons sélectionné de manière uniforme 2 milliards de colorations d'arêtes dans $\Phi(E)$ et $\Phi(E^c)$ respectivement. Pour chaque coloration, les ordres racines des arêtes ont été calculés. Seul les colorations de graphes à double diversité ont été comptées pour la comparaison. Comme illustration, les caractéristiques de colorations de graphes aléatoires à double diversité pour C_{P1} sont tracées Figure 16 où les estimations numériques des distributions de probabilités sont comparées à la coloration obtenue avec DECA.



(a) (b)

Figure 16: Distribution de $\eta(\phi)$ (figure a) et $\rho_{max}(\phi)$ (figure b) pour des colorations d'arêtes aléatoires à double diversité uniformément distribuées dans $\Phi(E)$ et $\Phi(E^c)$. Code produit $[12, 10]^{\otimes 2}$.

Performances des codes produits à double diversité

Les performances du décodage itératif de $C_P = C_1 \otimes C_2$ sont étudiées en présence d'effacements sur le canal avec ou sans coloration d'arêtes. Le décodeur itératif fait des itérations ligne-colonne. Le décodeur des codes élémentaires C_i peut être un décodeur algébrique remplissant les effacements (limités à $d_i - 1$ par ligne et colonne) ou un décodeur à maximum de vraisemblance. Comme démontré dans la Section 4.3.1 de la thèse, les ensembles d'arrêt de type II et de type III sont identiques car les codes élémentaires non binaires C_1 et C_2 sont MDS. La probabilité d'erreur par mot du décodeur itératif est notée par $P_{ew}^{\mathcal{G}}$. La probabilité d'erreur par mot du décodeur ML est notée P_{ew}^{ML}.

Effacements par blocs

Considérons le canal à effacement par blocs $CEC(q, \epsilon)$. Les N symboles d'un mot de code sont partitionnés en M blocs, chaque bloc contient des symboles associés aux arêtes de même couleur dans $\mathcal{G}$. Le canal $CEC(q, \epsilon)$ efface un bloc avec une probabilité ϵ. Le bloc est correctement reçu avec une probabilité $1-\epsilon$. Les effacements sont indépendants d'un bloc à l'autre. Nous disons qu'une *couleur est effacée* si le bloc associé aux N/M symboles est effacé. Supposons que $\mathcal{G}$ soit doté d'une coloration d'arêtes à double diversité ϕ (i.e. $L(\phi) = 2$). Ainsi, sur le canal à effacement par blocs $CEC(q, \epsilon)$, pour un rendement R satisfaisant

$$1 - \frac{2}{M} < R \leq 1 - \frac{1}{M}, \tag{17}$$

nous avons

$$\epsilon^2 \;\leq\; P_{ew}^{ML} \;\leq\; P_{ew}^{\mathcal{G}} \;\leq\; \sum_{i=2}^{M} \binom{M}{i} \epsilon^i (1-\epsilon)^{M-i}. \tag{18}$$

Comme ϕ a une double diversité, il existe deux couleurs parmi les M couleurs telles que le décodeur itératif va échouer si ces deux couleurs sont effacées. Ceci explique la borne supérieure de $P_{ew}^{\mathcal{G}}$ dans (18). Cette borne supérieure est valide pour tout rendement inférieur au rendement maximal atteignable pour obtenir une double diversité, i.e. $1-\frac{1}{M}$. Si nous choisissons un rendement $R \leq 1-\frac{2}{M}$, le décodeur ML peut atteindre une diversité $L = 3$. Or nous limitons cette étude à une diversité $L = 2$, il n'est donc pas judicieux de choisir un rendement $R \leq 1 - \frac{2}{M}$. Par conséquent, si le rendement est choisi dans l'intervalle $1 - \frac{2}{M} < R \leq 1 - \frac{1}{M}$, le décodeur ML de C_P peut seulement atteindre une diversité $L = 2$ et donc il existe une paire de couleurs qui une fois effacée ne peut pas être retrouvée par le décodeur ML.

Effacements indépendants

Considérons maintenant le canal à effacement i.i.d. $SEC(q, \epsilon)$. Les N symboles d'un mot de code sont indépendamment effacés par le canal. Un symbole est effacé avec une probabilité ϵ et est correctement reçu avec une probabilité $1-\epsilon$. Sur le canal $SEC(q, \epsilon)$, la coloration d'arêtes n'a pas d'effet sur les performances de C_P. Avant d'étudier les

performances du $SEC(q, \epsilon)$, nous formulons la proposition suivante et son corollaire.

Proposition *Soit $C_P = C_1 \otimes C_2$ un code produit à codes élémentaires non MDS. Tous les ensembles d'arrêt évidents (obvious) sont le support d'un mot de code du code produit.*

Corollaire *Considérons un code produit $C_P = C_1 \otimes C_2$ avec des codes élémentaires MDS non binaires. Supposons que les symboles de C_P soient transmis à travers un canal $SEC(q, \epsilon)$, alors pour $\epsilon \ll 1$, la probabilité d'erreur satisfait $P_{ew}^{\mathcal{G}} \sim P_{ew}^{ML}$.*

Les motifs d'effacement peuvent être décomposés suivant la taille de l'ensemble d'arrêt couvert. Le coefficient $\Psi_i(\mathcal{G})$ devient $\Psi_i(\mathcal{G}) = \sum_{w=d_1 d_2}^{i} \Psi_{i,w}(\mathcal{G})$, où $\Psi_{i,w}(\mathcal{G})$ est le nombre de motifs de poids i couvrant un ensemble d'arrêt de taille w. Il est clair que $\Psi_{w,w}(\mathcal{G}) = \tau_w$. Pour un petit $i - w$, $\Psi_{i,w}(\mathcal{G})$ peut être approximé par $\sum_{\mathcal{A}} \binom{N-\mathcal{A}}{i-w} \tau_{w,\mathcal{A}}$, où $\tau_{w,\mathcal{A}}$ est le nombre d'ensemble d'arrêt de taille w ayant $|\mathcal{R}(\mathcal{S})| = \mathcal{A}$. Une évaluation numérique de $\Psi_i(\mathcal{G})$ est envisageable pour des codes très courts ($N \leq 25$), cela devient très difficile pour des codes de taille modérée et au-delà, e.g. $N = 144$ et $N = 224$ pour les codes $[12, 10]^{\otimes 2}$ et $[14, 12] \otimes [16, 14]$ respectivement.

Pour $P_{ew}^{\mathcal{G}}$, grâce aux Théorèmes 4.2 et 4.3 (voir les détails de l'énumération dans la thèse), une borne de l'union peut facilement être établie. En effet, nous avons

$$P_{ew}^{\mathcal{G}} = Prob(\exists \mathcal{S} \ covered) \leq \sum_{w} Prob(\exists \mathcal{S} : |\mathcal{S}| = w, \mathcal{S} \ covered),$$

conduisant à

$$P_{ew}^{\mathcal{G}} \leq P^U(\epsilon) = \sum_{w=d_1 d_2}^{N} \tau_w \epsilon^w. \tag{19}$$

D'après le Théorème 4.2, la borne de l'union $P^U(\epsilon)$ pour le code produit $[12, 10, 3]_q^{\otimes 2}$ est

$$\begin{aligned} P^U(\epsilon) =& 48400\epsilon^9 + 6098400\epsilon^{12} + 23522400\epsilon^{13} + 17641800\epsilon^{14} \\ &+ 1754335440\epsilon^{15} + 9126691200\epsilon^{16} + o(\epsilon^{16}). \end{aligned}$$

Les performances de ce code sur le canal $SEC(q, \epsilon)$ sont illustrées Figure 17. Nous considérons le corps fini standard de taille $q = 256$. La borne de l'union pour la probabilité d'erreur par symbole $P_{es}^{\mathcal{G}}$ est obtenue en pondérant le terme de sommation dans (19) avec w/N, i.e. $P_{es}^{\mathcal{G}} \leq \sum_{w=d_1 d_2}^{N} \frac{w}{N} \tau_w \epsilon^w$. Comme on peut l'observer sur la Figure 17, la borne de l'union est suffisamment précise. De plus, les performances du décodeur algébrique itératif ligne-colonne sont très proches des performances du décodeur ML. Pour de faibles ϵ, les courbes se superposent comme prédit par le corollaire précédent.

Effacements à probabilité inégale

Dans les systèmes de communication et de stockage, des effacements à probabilité inégale peuvent se produire. Afin d'observer l'effet d'une coloration à double diversité sur la

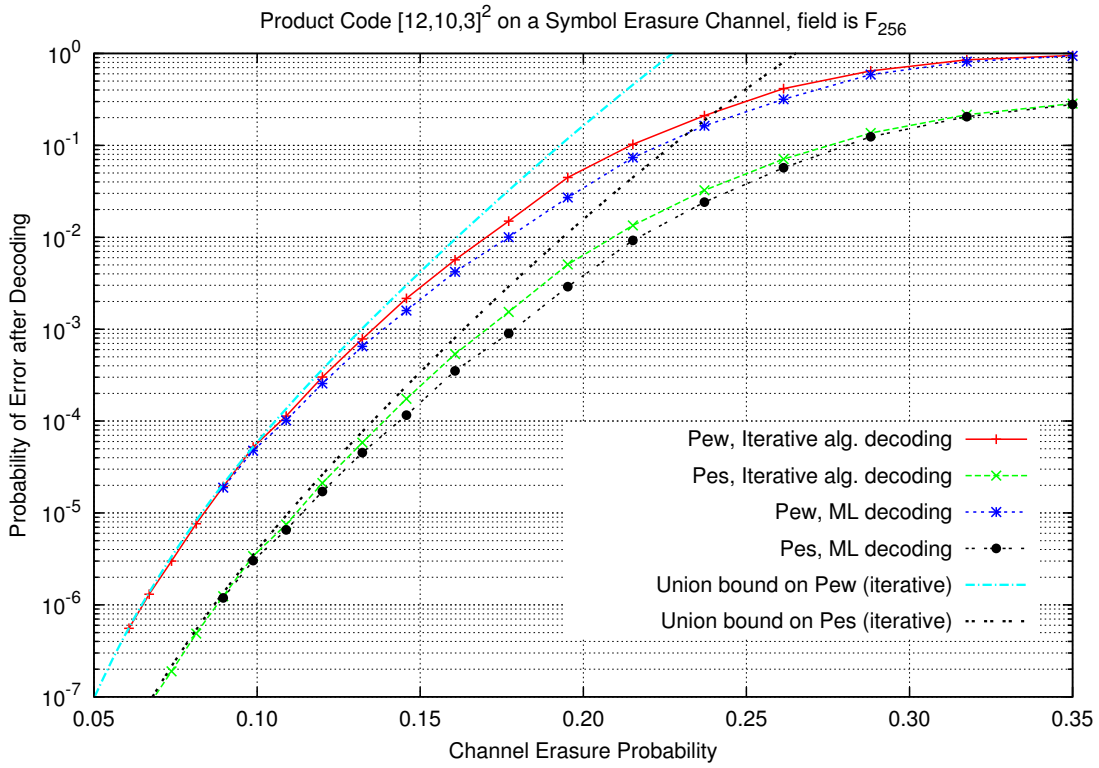


Figure 17: Code produit $[12, 10]_q^{\otimes 2}$, sans coloration d'arêtes. Performances des probabilités d'erreur par mot et par symbole pour un décodage itératif versus sa borne de l'union et le décodage ML.

performance d'un canal à effacement multiple, nous définissons le $SEC(q, \{\epsilon_i\}_{i=1}^M)$. Sur ce canal, les effacements de symboles sont indépendants mais la probabilité d'effacer un symbole est ϵ_i si il est associé à une arête dans $\mathcal{G}$ de couleur $\phi(e) = i$.
La borne de l'union peut facilement être modifiée pour obtenir

$$P_{ew}^{\mathcal{G}} \leq P^U(\epsilon_1, \ldots, \epsilon_M), \tag{20}$$

où

$$P^U(\epsilon_1, \ldots, \epsilon_M) = \sum_{w=d_1 d_2}^{N} \sum_{\substack{w_1, \ldots, w_M \\ : \sum_i w_i = w}} \tau(w_1, \ldots, w_M) \prod_{i=1}^{M} \epsilon_i^{w_i}. \tag{21}$$

Le coefficient $\tau(w_1, \ldots, w_M)$ est le nombre d'ensemble d'arrêt de taille $w = \sum_{i=1}^M w_i$, où i symboles ont la couleur i, $i = 1 \ldots M$. Clairement les coefficients $\tau(w_1, \ldots, w_M)$ dépendent de la coloration d'arêtes ϕ. Pour les colorations à double diversité et une palette de couleurs $M \geq 2$, ces coefficients satisfont la propriété suivante: les compositions faibles de l'entier w sont interdites par ϕ.

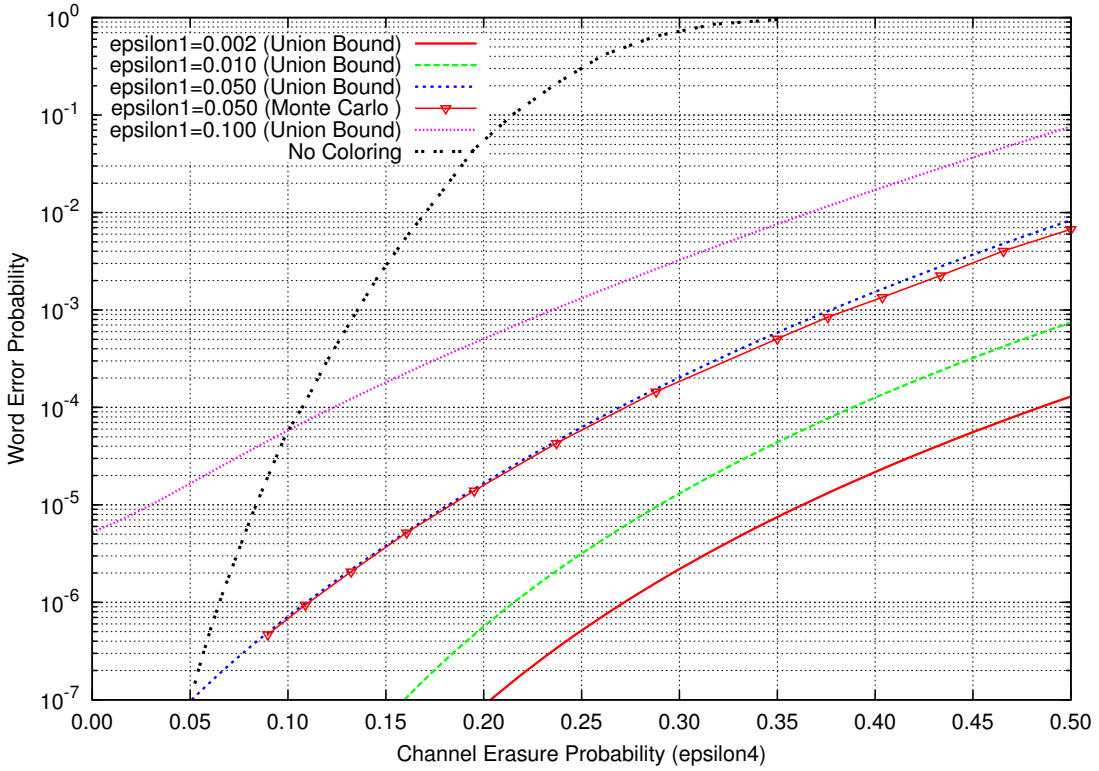


Figure 18: Code produit $[12, 10]_q^{\otimes 2}$ avec une coloration d'arêtes à double diversité. Probabilité d'erreur par mot versus ϵ_4, pour un décodage itératif sur le canal $SEC(q, \{\epsilon_i\}_{i=1}^4)$ avec $\epsilon_1 = \epsilon_2 = \epsilon_3$.

Par conséquent, le code produit devrait être performant si l'un des ϵ_i est proche de 1 et si le reste des ϵ_i sont suffisamment petits. Ce cas extrême est vrai grâce à la double diversité qui donne $P^U(0^{M-1}, 1^1) = 0$, où $(0^{M-1}, 1^1)$ représente tous les vecteurs avec toutes les composantes à 0 sauf en une position fixée à 1. La Figure 18 montre la performance du code produit $[12, 10]_q^{\otimes 2}$ sur le canal $SEC(q, \{\epsilon_i\}_{i=1}^M)$ avec $M = 4$ couleurs. La coloration d'arêtes considérée est celle produite par l'algorithme DECA de la Figure 15. L'expression de $P^U(\epsilon_1, \ldots, \epsilon_M)$ est déterminée par l'énumération des ensembles d'arrêt voir les Théorèmes 4.2 et 4.3. Le cas particulier où $\epsilon_1 = \epsilon_2 = \epsilon_3$ est considéré et les performances sont tracées en fonction de ϵ_4. Pour ϵ_1 fixé, la double diversité améliore considérablement les performances en fonction de ϵ_4.

Dans ce chapitre, les codes produits non binaires avec des codes élémentaires MDS ont été étudiés dans le cas d'un décodage algébrique itératif ligne-colonne. Les canaux avec des effacements indépendants et par blocs ont été considérés. Les ensembles d'arrêt sont définis dans le contexte des codes élémentaires MDS et une relation a été établie

avec la représentation graphique du code produit. Une caractérisation complète des ensembles d'arrêt est donnée jusqu'à un poids $(d_1 + 1)(d_2 + 1)$. Nous avons ensuite proposé un algorithme à évolution différentielle pour concevoir une coloration d'arêtes avec un grand nombre de symboles ayant un ordre racine minimal. La complexité de cet algorithme par itération est $o(M^{\aleph})$, où $\aleph$ est le paramètre à évolution différentielle. Les performances d'un code produit à codes élémentaires MDS avec et sans coloration d'arêtes à double diversité ont été analysées. De plus, il a été prouvé que pour de petites probabilités d'effacement sur le canal, les décodeurs itératif et ML coïncident.

Conclusions

Les travaux de cette thèse sont centrés sur le stockage distribué dans les centres de données en interaction avec les ressources du cloud computing. Pour le stockage distribué, le modèle du canal à effacement est combiné aux structures de codes particulières. Nous avons largement utilisé le modèle du canal à effacement dans ces travaux de recherche. Le canal à effacement est un modèle simple, adapté à la fois aux longueurs finies et à l'analyse de codes de longueurs infinies. Malgré sa simplicité, il permet une bonne compréhension du comportement des codes et aide à leur conception.

La première partie de ce travail est consacrée à l'étude des performances des codes spatialement couplés sur le canal à effacement. Le couplage spatial des codes en graphe, principalement les codes LDPC, est une méthode récemment développée pour atteindre la capacité sur tout canal symétrique sans mémoire à entrée binaire. La capacité de Shannon est approchée en laissant trois paramètres devenir suffisamment grands: la distribution de degrés, la taille de la fenêtre de couplage et la longueur de la chaîne. Dans notre travail, nous avons montré qu'une petite fenêtre de couplage donne des résultats impressionnants en termes de seuils du décodage itératif. Cette nouvelle méthode est appelée couplage spatial multicouches à sens direct. Cette méthode est inspirée du codage par couches et a la mémoire la plus courte possible, i.e. $w = 2$. Les arêtes des ensembles locaux et ceux définissant le couplage spatial sont construites séparément. Cette nouvelle méthode permet ainsi la construction de chaînes de couplage non uniformes et spatialement variantes.
Ensuite, nous avons appliqué le couplage spatial multicouches à sens direct aux codes Root-LDPC. Dans ce cas, seuls les bits de parité ont été couplés. Le couplage spatial des bits de parité joue le rôle de dopage pour les codes à diversité pleine. L'étude des performances sur canal à effacement a montré une amélioration de la saturation de la frontière de coupure de code par rapport aux codes non couplés. Les ensembles Root-LDPC spatialement couplés atteignent des seuils très proches de la limite de capacité dans le plan à effacement. Ce comportement dans le plan à effacement entraîne automatiquement une saturation de la frontière de coupure du code Root-LDPC sur un canal à évanouissements par blocs avec un bruit additif.
Les ensembles Root-LDPC spatialement couplés sont appliqués aux systèmes de stockage distribué et de diversité. De la structure à double diversité des codes Root-LDPC, il

est évident que les bits d'information ont une localité $d_c - 1$. Dans le cas particulier des effacements par blocs/couleurs, la localité d'un ensemble LDPC aléatoire est $O(\log(n))$ alors que celle d'un code Root-LDPC est maintenue à $d_c - 1$.

Dans la deuxième partie de cette thèse, les codes produits non binaires avec des codes élémentaires MDS sont étudiés dans le contexte d'un décodage itératif algébrique ligne-colonne. Des canaux avec des effacements indépendants et en blocs sont considérés. Les concepts de rootcheck et de colorations d'arêtes à double diversité sont décrits après l'introduction d'une représentation graphique compacte des codes produits. De nouveaux résultats sur le décodage itératif des codes produits sont obtenus. Pour retrouver les symboles effacés, une borne supérieure du nombre d'itérations de décodage est donnée en fonction de la taille du graphe et la taille M de la palette de couleurs. Les ensembles d'arrêt sont définis dans le contexte de codes élémentaires MDS et une relation est établie avec la représentation graphique du code produit. Une caractérisation complète des ensembles d'arrêt est donnée jusqu'à un poids $(d_1+1)(d_2+1)$. La représentation graphique compacte nouvellement introduite des codes produits nous a permis de proposer un algorithme de coloration d'arêtes à évolution différentielle. Cet algorithme permet de concevoir des colorations avec une grande population de symboles d'ordre minimal. La complexité de cet algorithme par itération est $o(M^{\aleph})$, où $\aleph$ est le paramètre à évolution différentielle. Les performances des codes produits avec et sans coloration à double diversité sont analysées. Il est aussi montré que pour de faibles probabilités d'effacement, les performances du décodeur itératif coïncident avec celles du décodeur ML. Ceci est une belle propriété des codes produits à codes élémentaires MDS.

Introduction

Current technology in both software and hardware, combined to mathematical tools driven by Coding Theory, Information Theory, and Communication Theory, is dramatically changing our society. These tools helped wireless networking technologies to make a spectacular progress and in turn to become an almost ubiquitous component of modern life revolutionizing education, culture and the exchange of information in general. New efficient techniques appeared recently for point-to-point communications and for information transmission and processing in networks, namely: network coding, coding for distributed storage, spatially-coupled graph codes, and lattices in high dimension for iterative decoding.

In nowadays information systems, Cloud Computing is a common concept where most of the resources are available online. Such a concept is of great interest because resources are evolving in an instantaneous and automatic manner. There exists in communication systems a sub-class of Cloud Computing referred to as Distributed Computing. In the latter, many network nodes proceed jointly for solving a problem or for answering a request. Distributed Storage is yet another case of Distributed Computing, where a file is stored in parts on multiple network nodes.

This Ph.D research work investigates distributed storage over data centers interacting with cloud computing resources, serving clients or mobile users. This case can be represented by the canonical scenario depicted in Figure 19 where a source, requesting cloud services, disperses a file across n nodes with enhanced computing and storage capabilities. Error-control codes are the main tool for distributive computing/storage. Besides trivial repetition coding, a Reed-Solomon code or any Maximum Distance Separable (MDS) code is capable of reconstructing the source file [54]. The main objective of distributed coding for storage is to increase the reliability of reconstruction and minimize the required bandwidth.

Coding can be used to offer different levels of error protection for the requested quality of service and power constraints depending on the wireless channel state between the

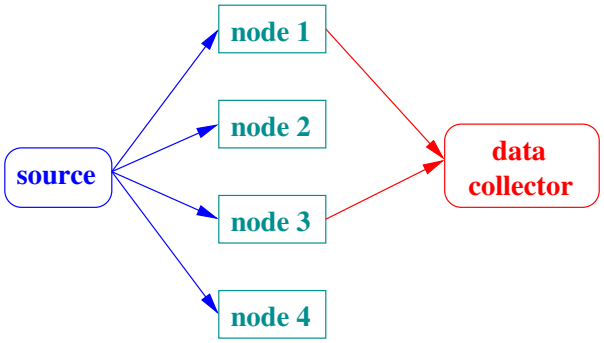


Figure 19: Distributed storage system with a file distributed on $n = 4$ nodes and collected from $k = 2$ nodes.

k nodes and the terminal. Such unequal error protection (UEP) coding schemes have been studied for wireless networks based on Low-Density Parity-Check (LDPC) codes, e.g. see [73]. On the other hand, upon failure of a node within the network, coding can also be used to regenerate the failed nodes. During the repair process, the replacement node can contain the same information or it may contain a functionally equivalent information, while minimizing the repair bandwidth, i.e. minimizing the number of network downloads to replace the failing node.

Many coding families are potential candidates for distributed storage, most of them are concatenated codes. This thesis consider LDPC structures and product codes structures. An LDPC is the concatenation of a high number of small check nodes. On top of this concatenation, spatial coupling involves multiple copies of the same LDPC ensemble where spatial edges connect one copy to another. On the other hand, a product code is a bi-dimensional code built from row and column code components. Its matrix representation has a complete graph equivalent. Product codes are also concatenated codes and they may be considered as cousins to LDPC thanks to their iterative row-column decoding.

Thesis Outline and Contributions

This dissertation is composed of four main chapters. The contents and the contributions of each one of them are summarized in the following.

Chapter 1: Chapter 1 lists important error-correcting codes as found in the current literature. First, the basic properties of the binary erasure channel are given. Then, a review of the fundamental features is given for Reed-Solomon codes, Product codes, LDPC, Fountain codes, spatially-coupled codes, Polar codes, and finally Reed-Muller Codes. Readers who are experts in coding theory may skip this chapter.

Chapter 2: We aim in this chapter to present distributed storage constraints and make an overview of existing codes for such storage systems. Based on the study of a practical case of a Facebook data center, the constraints of coding for distributive storage are presented in Section 2.1. In nowadays systems, coding for distributed storage systems is achieved using two main approaches: Replication and Erasure codes. These two methods are analyzed and discussed in Section 2.2. In the literature, other coding techniques have been proposed to overcome the repair process drawbacks of MDS erasure codes, namely regenerating codes and locally repairable codes. They are described in Sections 2.3 & 2.4. In the context of distributed storage, sparse-graph codes, such as Fountain codes and LDPC codes, can also be used after a convenient arrangement. Section 2.5 is dedicated to the study of Fountain codes. In this last section, we have examined the properties of repairable Fountain codes in order to determine the necessary criteria to design efficient repair codes for distributed storage systems.

Chapter 3: Chapter 3 reviews in first Section 3.1 uniform spatial coupling as known in the literature. Then, a new method for spatial coupling of low-density parity-check ensembles is proposed in Section 3.2. The method is inspired from overlapped layered coding. Edges of local ensembles and those defining the spatial coupling are separately built. The new method allows, in Section 3.2, the construction of non-uniform coupling chains with near-Shannon spatially-varying thresholds under iterative decoding. The space-varying coupling is formed by gluing non-identical LDPC ensembles. The binary erasure channel (BEC) is the default transmission channel, however, all results are extendable to binary memoryless symmetric (BMS) channels.

Random low-density parity-check (LDPC) ensembles do not achieve full diversity on non-ergodic channels. To cope with block erasures and quasi-static fading, a special Root-LDPC structure is considered. A Root-LDPC ensemble guarantees that any information bit receives messages from all channel states. Results in the literature show that the gap between the Root-LDPC boundary and the capacity boundary in the fading plane is not small enough. We propose in Section 3.3 to saturate the whole Root-LDPC boundary via spatial coupling. For simplicity, we adopt an equivalent erasure channel model rather than a block fading model. It is shown that spatial coupling of parity bits is sufficient to saturate the Root-LDPC threshold boundary in the erasure plane.

Finally, in Section 3.4, spatial coupling is applied on parity bits of a Root-LDPC ensemble designed for a channel with 4 block-erasure states and a maximal design rate of 3/4 attaining double diversity. The advantage of spatial coupling is shown in the erasure plane as an improvement of a threshold boundary, under independent erasures. The spatial coupling maintains the double diversity because it connects parity bits only. The drawback of this partial coupling is a weak saturation of the threshold boundary towards the capacity boundary. This method is proposed as the design of distributed coding repair schemes.

Chapter 4: We consider non-binary product codes with MDS components and their iterative row-column algebraic decoding on the erasure channel. Both independent and block erasures are considered in this chapter. A compact graph representation is introduced on which we define double-diversity edge colorings via the rootcheck concept. An upper bound of the number of decoding iterations is given as a function of the graph size and the color palette size M. Stopping sets are defined in the context of MDS components and a relationship is established with the graph representation. A full characterization of these stopping sets is given up to a size $(d_1+1)(d_2+1)$, where d_1 and d_2 are the minimum Hamming distances of the column and row MDS components respectively. Then, we propose a differential evolution edge coloring algorithm that produces colorings with a large population of minimal rootcheck order symbols. The complexity of this algorithm per iteration is $o(M^{\aleph})$, for a given differential evolution parameter $\aleph$, where $M^{\aleph}$ itself is small with respect to the huge cardinality of the coloring ensemble. The performance of MDS-based product codes with and without double-diversity coloring is analyzed in presence of both block and independent erasures. In the latter case, ML and iterative decoding are proven to coincide at small channel erasure probability. Furthermore, numerical results show excellent performance in presence of unequal erasure probability due to double-diversity colorings.
The main results in this chapter are:

- Establishing a new compact graph for product codes. The compact graph has many advantages, the main one being its ability to imitate a Tanner graph with parity-check nodes. The compact graph is also the basis for the differential evolution edge coloring. See Section 4.2.2.
- Iterative decoding analysis of finite-length product codes, mainly the proof of new bounds on the number of decoding iterations. See Theorem 4.1 and Corollary 4.1.
- Proving new properties of stopping sets for product codes with MDS components. See Propositions 4.1&4.2, Corollaries 4.2-4.4, and Lemmas 4.1&4.2.
- Complete enumeration and characterization of stopping sets up to a size $(d_1+1)(d_2+1)$, where d_1, d_2 are the minimum Hamming distances of the component codes. This stopping set enumeration goes beyond the weight $d_1d_2+\max(d_1,d_2)$ of Tolhuizen's Theorem 3 for codeword enumeration in the MDS components case. See Lemmas 4.3&4.4 and Theorems 4.2&4.3.
- A new edge coloring algorithm (DECA) capable of producing double-diversity colorings despite the huge size of the coloring ensembles. See Section 4.4.2.
- Construction via the DECA algorithm of product codes maximizing the number of edges with root order 1, i.e. minimizing the locality when the process of repairing nodes is considered. See Section 4.4.3.
- First numerical results for MDS-based product codes on erasure channels showing how close iterative decoding is to ML decoding, mainly for small ϵ. We proved

that iterative decoding perform as well as ML decoding (the ratio of error probabilities tends to 1) for MDS-based product codes at small ϵ. See Proposition 4.3, Corollary 4.5, and other performance results in Section 4.5.2.

- Great advantage of double-diversity colorings of product codes (with respect to codes without coloring) in presence of unequal probability erasures. Thus, double-diversity colorings are efficient on both ergodic and non-ergodic erasure channels. See Section 4.5.3.

Finally, the results of this work and some future research perspectives are summarized in a general conclusion.

Chapter 1

Channel Coding Background and Recent Advances

In the last decade, coding theory evolved from algebraic doing to modern coding. Nowadays, in recent coding, all areas are interconnected and any separation between algebraic-based and graph-based codes is senseless. Besides the algebraic description of Reed-Solomon codes (and BCH codes in general), the multi-facet description of Reed-Muller codes, and the graph representation of low-density parity-check codes, polar codes have a new description importing its tools from information theory. This chapter briefly describes some important error-correcting codes that are directly related to our main chapters in this report dealing with spatial coupling of LDPC codes and RS-based product codes. In addition, some codes are quickly reviewed as they correspond to some hot areas in current research.

1.1 Erasure channel essential

Erasure channel is the simplest non-trivial channel model. This model has for each transmitted input two possible outcomes: either it is correctly received or it is known to be lost, no corruption is possible. The decoding problem is to find the values of the information erased given the locations of erasures and the non-erased part of the codeword. The transmitter and the receiver know the time t, i.e. they are both synchronized. The channel input at time t is denoted by X_t and the corresponding output Y_t.

The Binary Erasure Channel (BEC) was introduced by Peter Elias in 1954 as a toy example. In the case of the binary erasure channel the channel input is binary and $X_t \in \{0, 1\}$. The corresponding output Y_t is ternary and takes on values in the alphabet $\{X_t, ?\}$, where ? indicates an erasure. One transmitted bit is erased with probability ϵ, i.e. $P(Y_t =?) = \epsilon$ and $P(Y_t = 0) = P(Y_t = 1) = 1 - \epsilon$. Erasure occurs for each t independently, therefore the channel is said to be memoryless. The Binary Erasure Channel, BEC(ϵ) is illustrated in Figure 1.1.

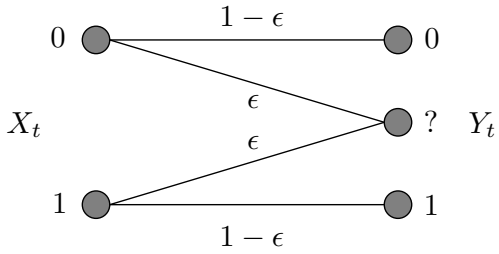


Figure 1.1: Binary erasure channel with parameter ϵ: BEC(ϵ).

In this thesis, we consider also non-binary erasure channel. For this channel model, the input X_t is q-ary and belongs to the finite field $\mathbb{F}_q$. The integer q is the number of elements of the finite field $\mathbb{F}_q$. Respectively, Y_t belongs to $\mathbb{F}_q \cup \{?\}$. We can notice that binary erasure channel is a particular case where $q = 2$.

Our interest is focused on three erasure channels:

• The symbol erasure channel: $SEC(q, \epsilon)$. The N symbols of a codeword are independently erased. A symbol is erased with a probability ϵ and is correctly received with a probability $1 - \epsilon$. The erasure events are independent from one symbol to another. This channel is said to be ergodic. Ergodicity of a channel is defined as follows:

Definition 1.1. A channel is said to be ergodic if the time average of the signal is equal to the collective average. In other words, the randomness of the channel gain can be averaged out over time.

In a similar way we can define a non-ergodic channel:

Corollary 1.1. *A channel is non-ergodic if its channel gain is a random variable and is assumed to be invariant for some time period. The channel gain process is stationary but not ergodic, i.e., the time average is not equal to the ensemble average. In other words, the randomness of the channel gain cannot be averaged out over time. So long-term constant bit rates cannot be supported.*

• The block erasure channel: $CEC(q, \epsilon)$. The N symbols of a codeword are partitioned into M blocks, each block contain symbols. M is the number of degrees of freedom of the channel. The channel erases a block with a probability ϵ. The block is correctly received with a probability $1 - \epsilon$. The erasure events are independent from one block to another. From Corollary 1.1, the block erasure channel is a non-ergodic channel. In this context, non-ergodicity means that the transmitted codeword spans only a finite number M of independent realizations of the channel independently from its length. Block erasure channel is a particular case of non-ergodic (quasi-static) fading channels, where the fading coefficient α belongs to $\{0, +\infty\}$.

• The unequal probability symbol erasure channel: $SEC(q, \{\epsilon_i\}_{i=1}^{M})$. As in the previous case, the N symbols of a codeword are partitioned into M blocks. The particularity of this channel is that the symbol erasures are independent but their probability varies from one block to another. Symbol erasures occur independently, consequently Definition1.1 leads to conclude that this channel is ergodic.

In the case of ergodic erasure channel, we can define the channel capacity as follows:

$$C_{BEC}(q, \{\epsilon_i\}_{i=1}^{M}) = \log(q) \times (1 - \frac{1}{M} \sum_{i=1}^{M} \epsilon_i) \text{ bits per channel use.} \tag{1.1}$$

1.2 Linear block codes

The aim of this section is to present briefly the main properties of linear block codes. A block code $[n, k, d]_q$ encodes the input data stream by dividing it into k blocks of data symbols. The encoder takes each k information word and adds parity symbols to make an n symbol codeword which is transmitted over the channel. There are $n - k$ parity symbols. k is known as the code dimension and n is known as the block-length.

Definition 1.2. An $[n, k, d]_q$ q-ary block code C is a set of q^k q-ary sequences of length n, called codewords.

Definition 1.3. The coding rate R of a block code is the ratio between its dimension k and its block-length n. It is defined as

$$R = \frac{k}{n} \tag{1.2}$$

Let $\mathbf{x}$ and $\mathbf{y}$ be two codewords of C.

Definition 1.4. The Hamming distance $d(\mathbf{x}, \mathbf{y})$ between $\mathbf{x}$ and $\mathbf{y}$ is defined as the number of positions at which they differ.

Definition 1.5. The minimum Hamming distance d is given by

$$d = \min_{\mathbf{x} \neq \mathbf{y}} d(\mathbf{x}, \mathbf{y}), \text{ where } \mathbf{x}, \mathbf{y} \in C. \tag{1.3}$$

Linear block codes are subject to the additional constraint that any linear combination of the codewords is also a codeword. In the q-ary case, the addition and the scalar multiplication are component-wise modulo q.

Definition 1.6. For a linear block code $[n, k, d]_q$, the code dimension k is a vector subspace of $\mathbb{F}_q^n$.

Definition 1.7. The minimum Hamming distance d of any linear block code satisfies the Singleton bound:

$$d \leq 1 + n - k. \tag{1.4}$$

Definition 1.8. Any code whose minimum Hamming distance d satisfies:

$$d = 1 + n - k, \tag{1.5}$$

is called a Maximum-Distance Separable (MDS) code.

In the following sections of this chapter, linear block codes are briefly presented. Linear block codes can be introduced in three different ways:

- Polynomial representation, e.g. generator polynomial, check polynomial.
- Matrix representation, e.g. generator matrix, parity-check matrix.
- Graphical representation, e.g. Tanner graph, factor graph.

A convenient representation will be chosen according to the mathematical tool needed for the introduction of the code.

1.3 Reed-Solomon codes

Reed–Solomon (RS) codes are block-based error-correcting codes that were introduced by Irving S. Reed and Gustave Solomon in 1960 [84]. They are applied in all areas requiring reliable data. The most prominent applications are in digital communications and storage. These codes were defined for noisy channel where errors, i.e. the data transmitted is modified during the transmission, occur during transmission. Reed-Solomon codes can also be used to correct burst of errors. A burst of errors is a series of bits in the codeword which are received in error.

RS codes are linear block codes and belong to the family of BCH (**B**ose, Ray-**C**haudhuri et **H**ocquenghem) codes, which are a class of cyclic error-correcting codes constructed using finite fields. For more details see [46], [12], [10], [65].
An RS code is a cyclic linear block code over the finite field $\mathbb{F}_q$ of length n, with n a divisor of $q-1$. When $n = q-1$ the RS code is said to be primitive. Reed-Solomon codes are maximum distance separable codes as they attain the Singleton bound in Definition 1.8. In the sequel of this thesis, we only consider non-trivial non-binary RS codes where $q > n > 2$.

Let ν be the number of errors and μ the number of erasures occurred during the transmission. In presence of errors only, any pattern of ν errors can be decoded, provided that

$$\nu \leq \left\lfloor \frac{d-1}{2} \right\rfloor. \tag{1.6}$$

Similarly, in presence of erasures only, any pattern of μ erasures can be decoded, provided that

$$\mu \leq d - 1. \tag{1.7}$$

In presence of both errors and erasures the condition of successful decoding is given by:

$$2\nu + \mu \leq d - 1 \tag{1.8}$$

The following list sums up several advantages of Reed-Solomon codes:

- RS codes are useful when a code is required of length less that the size of the field.
- They have the highest possible minimum distance, i.e. they are MDS.
- They allow to correct a burst of errors.
- They are convenient for constructing other codes, as we shall see in Section 1.4.

Reed-Solomon encoder

The BCH view: The codeword as a sequence of coefficients

Consider an $[n, k, d]_q$ Reed-Solomon code. A codeword $c(x)$ is generated using a special polynomial called generator polynomial and denoted $g(x)$. Codewords are valid if they are exactly divisible by the generator polynomial. Let α be a primitive root of $\mathbb{F}(q)$. The general form of the generator polynomial is given by:

$$g(x) = (x - \alpha^b)(x - \alpha^{b+1}) \dots (x - \alpha^{b+d-2}). \tag{1.9}$$

Usually, but not always, $b = 1$.
Reed-Solomon encoder can be systematic or non-systematic.

Definition 1.9. Let C be a $[n, k, d]_q$ linear or non-linear code. The encoder is called systematic if the information data is embedded in the encoded output. An non-systematic encoder scrambles the message, i.e. the encoded message does not contain the input symbols.

In the case of non-systematic encoder, further computations are necessary to recover the information data.
A non-systematic codeword is constructed using:

$$c(x) = g(x)i(x), \tag{1.10}$$

where $i(x)$ is the information polynomial. In the case of systematic codeword, the previous equation is modified as follows:

$$c(x) = i(x)x^{n-k} + [i(x)x^{n-k}] \bmod(g(x)), \tag{1.11}$$

where $[i(x)x^{n-k}] \bmod(g(x))$ denotes the parity polynomial.

Reed & Solomon's original view: The codeword as a sequence of values

This method is the original one proposed by Reed and Solomon. Let $\mathbf{i} = (i_0, i_1, \ldots, i_{k-1})$, $i_j \in \mathbb{F}_q$, be the message to be encoded. Let define $i(x)$ such as:

$$i(z) = \sum_{j=0}^{k-1} i_j z^j. \tag{1.12}$$

The codeword is obtained by evaluating i at n different points $\alpha^0, \alpha^1, \ldots, \alpha^{n-1}$ of the finite field $\mathbb{F}_q$. Thus,

$$\mathbf{c} = (i(1), i(\alpha), \ldots, i(\alpha^{n-1})). \tag{1.13}$$

We show that $\mathbf{c}$ is a RS codeword by verifying that $c(x) = \sum_{j=0}^{n-1} c_j x^j$ has $\alpha, \alpha^2, \ldots, \alpha^{n-1}$ as zeros.

Reed-Solomon decoder

Another advantage of RS codes is the ease with which they can be decoded, namely, via an algebraic method known as syndrome decoding.
Consider an RS code of parameters $[n, k, d]_q$. Let $c(x)$ be the transmitted codeword and $r(x)$ be the received codeword. The received codeword for an error-only channel can be expressed as

$$r(x) = c(x) + e(x), \tag{1.14}$$

where $e(x)$ is an unknown error polynomial. The error polynomial $e(x)$ is

$$e(x) = e_{n-1}x^{n-1} + e_{n-2}x^{n-2} + \ldots + e_1 x^1 + e_0, \tag{1.15}$$

where at most $\left\lfloor \frac{d-1}{2} \right\rfloor$ coefficients are nonzero. Suppose that ν errors occur at unknown position $i_1, i_2, \ldots, i_\nu$, then $e(x)$ can be written

$$e(x) = e_{i_1}x^{i_1} + e_{i_2}x^{i_2} + \ldots + e_{i_\nu}x^{i_\nu}, \tag{1.16}$$

where e_{i_l} is the unknown magnitude of the lth error. Note that the number ν of errors is also unknown. A Reed-Solomon decoder attempts to identify the number of errors, their positions and their values (magnitudes). Based on this, $e(x)$ can be calculated and subtracted from $r(x)$ to get the original message $c(x)$.

RS codes can be decoded either in time domain or in frequency domain. Decoding in the frequency domain, using Fourier transform techniques, can offer computational and implementation advantages. For this reason, decoding algorithms are presented in the frequency domain in the sequel and they are considered for channels with algebraic errors which is the most general case.

For erasures-only channels, the positions of erased symbols are known. Thus, the erasure-locator polynomial and its roots are already determined. In this case, one way

to decode is to guess the erased symbols and then apply any error-correction procedure. Reed-Solomon codes over erasures-only channels can be decoded via Maximum-Likelihood (ML) decoder. This is a non-iterative decoder based on a Gaussian reduction of the parity-check matrix of the RS code.
For a channel making both errors and erasures, the procedure is similar to the one with errors but erasure filling is incorporated into the algorithm. The reader should refer to chapter 7 in [10] for more details about these algorithms.
In presence of errors, the most common ones follow this general outline:

1. Calculate the syndromes S_j for the received vector.
2. Determine the number of errors and the error-locator polynomial $\Lambda(x)$.
3. Calculate the roots of the error-locator polynomial to find the error locations.
4. Determine the error-evaluator polynomial $\Gamma(x)$ and the error values.
5. Correct the errors.

Calculate the syndromes

The syndrome values are formed by evaluating $r(x)$ at $\alpha^{i_1}, \ldots, \alpha^{i_\nu}$. Thus the syndromes are

$$S_j = r(\alpha^j) = c(\alpha^j) + e(\alpha^j) = e(\alpha^j). \tag{1.17}$$

If there is no error, $S_j = 0$ for all $j = 1, \ldots, d-1$. If the syndromes are all zero, then the decoding is achieved.
We define the lth error values by $Y_l = e_{i_l}$, and the lth error location numbers by $X_l = \alpha^{i_l}$, for $l = 1, \ldots, \nu$. With these notations the syndrome values are given by

$$\begin{aligned} S_1 &= Y_1X_1 + Y_2X_2 + \ldots + Y_\nu X_\nu \\ S_2 &= Y_1X_1^2 + Y_2X_2^2 + \ldots + Y_\nu X_\nu^2 \\ &\vdots \\ S_{d-1} &= Y_1X_1^{d-1} + Y_2X_2^{d-1} + \ldots + Y_\nu X_\nu^{d-1} \end{aligned}$$

The decoding problem is now reduced to solve a system of non-linear equations. The solution to this system is unique. The set of non-linear equations is too difficult to solve directly, therefore a process called locator decoding is computed.

Find the symbol error locations

This step consists in determining the error-locator polynomial $\Lambda(x)$ in order to find the error locations. $\Lambda(x)$ is defined as

$$\Lambda(x) = \prod_{l=1}^{\nu}(1 - xX_l) = 1 + \Lambda_1 x + \Lambda_2 x^2 + \ldots + \Lambda_\nu x^\nu \tag{1.18}$$

The zeros of $\Lambda(x)$ are the reciprocals X_l^{-1}:

$$\Lambda(X_l^{-1}) = 0 \tag{1.19}$$

If the coefficients of $\Lambda(x)$ are known, then we can find its zeros to obtain the error locations.
We multiply Equation (1.18) by $Y_l X_l^{j+\nu}$ for $j = 1, \ldots, d-1$ and set $x = X_l^{-1}$:

$$Y_l(X_l^{j+\nu} + \Lambda_1 X_l^{j+\nu-1} + \ldots + \Lambda_\nu X_l^j) = 0. \tag{1.20}$$

Equation (1.20) is true for all l. Then we sum these equations from $l = 1$ to $l = \nu$. For each j, $j = 1, \ldots, d-1$, we have:

$$\sum_{l=1}^{\nu} Y_l(X_l^{j+\nu} + \Lambda_1 X_l^{j+\nu-1} + \ldots + \Lambda_\nu X_l^j) = 0, \tag{1.21}$$

from which we can deduce:

$$S_{j+\nu} + \Lambda_1 S_{j+\nu-1} + \Lambda_2 S_{j+\nu-2} + \ldots + \Lambda_\nu S_j = 0. \tag{1.22}$$

Finally, as $\nu \leq \left\lfloor \frac{d-1}{2} \right\rfloor$ if $1 \leq j \leq d-1-\nu$, the subscripts always specify known syndromes. The set of linear equations relating the syndromes to the coefficients is:

$$\Lambda_1 S_{j+\nu-1} + \Lambda_2 S_{j+\nu-2} + \ldots + \Lambda_\nu S_j = -S_{j+\nu}, \quad j = 1, \ldots, d-1-\nu. \tag{1.23}$$

This system can be written as

$$S_k = -\sum_{j=1}^{\nu} \Lambda_j S_{k-j}, \quad k = \nu+1, \ldots, d-1. \tag{1.24}$$

The connection weights $(\Lambda_1, \Lambda_2, \ldots, \Lambda_\nu)$ for the smallest $\nu \leq \left\lfloor \frac{d-1}{2} \right\rfloor$ for which the system (1.24) has a solution can be found with different algorithms. The most known are:

- Peterson algorithm.
- Berlekamp–Massey algorithm.

and are explained in chapters 6 and 7 in [10]. Peterson algorithm has a complexity proportional to ν^3 due to the inversion of a $\nu \times \nu$ matrix. Consequently, when ν is large another method should be used to find the polynomial Λ. The Berlekamp–Massey algorithm is an alternative procedure. It allows to solve the system of equations (1.24) but in a modified version. In this case, the system of equations has to be solved for the smallest $\nu \leq d-1$. This algorithm is iterative and has to create the minimum-length linear recursion $(\Lambda(x), \nu)$. At each iteration $r = 1, \ldots, d-1$, a discrepancy polynomial

$\Delta(x)$ based on a current instance of $\Lambda^{(r)}(x)$ with an assumed number of errors L_r is calculated. The discrepancy polynomial is defined as

$$\Delta(x) = \Lambda(x)S(x). \tag{1.25}$$

At iteration r, in order to compute the shortest linear recursion $(\Lambda^{(r)}, L_r)$ the earlier iterations are used in the discrepancy polynomial which is defined as,

$$\Delta_r = \sum_{j=1}^{L_{r-1}} \Lambda_j^{(r-1)} S_{r-j}. \tag{1.26}$$

Then, $\Lambda^{(r)}(x)$ and L_r are adjusted until Δ_r attains zero and then the next iteration is computed until $r = d - 1$.

At this point of the procedure the polynomial $\Lambda(x)$ is known, we have to determine its roots in order to find the error positions. Usually, because there is only a finite number of field elements to check, the simplest way to find the roots is by trial and error, for example by using the Chien search algorithm [10]. At the end of this step the error-locator polynomial can be written as:

$$\Lambda(x) = (1 - \alpha^{i_1}x)(1 - \alpha^{i_2}x)\dots(1 - \alpha^{i_\nu}x), \tag{1.27}$$

where the error positions $i_1, i_2, \dots, i_\nu$ are now known.

Calculate error magnitudes

In order to correct the errors the next step is to find the error values. One method is to compute the Gorenstein–Zierler algorithm after the Peterson procedure to find the error magnitudes. This algorithm is based on matrix inversions. The whole procedure is known as Peterson-Gorenstein–Zierler algorithm.
Another process depends on the error-evaluator polynomial $\Gamma(x)$. This polynomial is defined as:

$$\Gamma(x) = [\Lambda(x)S(x)] \bmod (x^d). \tag{1.28}$$

Using that

$$S(x) = \sum_{j=1}^{d-1} S_j x^j = \sum_{j=1}^{d-1} \sum_{l=1}^{\nu} Y_l X_l^j x^j,$$

Equation (1.28) becomes

$$\Gamma(x) = \sum_{i=1}^{\nu} xY_iX_i \prod_{j\neq i}(1 - X_jx). \tag{1.29}$$

The Forney algorithm, built on $\Gamma(x)$, can be used to find the error values. The lth error value is given by

$$Y_l = -X_l \frac{\Gamma(X_l^{-1})}{\Lambda'(X_l^{-1})}. \tag{1.30}$$

Using the error values and their locations, errors can be corrected by subtracting error values at error locations.

Another iterative method to find the error-locator polynomial and calculate error magnitudes is based on the Extended Euclidean algorithm. This algorithm computes the polynomial greatest common divisor of two univariate polynomials for locating least common divisor of polynomials $\Lambda(x)S(x)$ and x^{d-1}. The key idea of this algorithm is based on the following equation

$$\Lambda(x)S(x) = Q(x)x^{d-1} + \Gamma(x). \tag{1.31}$$

The goal is not to find the least common divisor. The extended Euclidean algorithm can find a series of polynomials of the form

$$A_r(x)S(x) + B_r(x)x^{d-1} = R_r(x). \tag{1.32}$$

At each iteration r until the degree of R_r is greater than $\left\lfloor \frac{d-1}{2} \right\rfloor$ the polynomials are updated for the next iteration as follows:

$$\begin{aligned} Q &= \frac{R_{r-2}}{R_{r-1}}, \\ R_r &= R_{r-2} - QR_{r-1}, \\ A_r &= A_{r-2} - QA_{r-1}. \end{aligned}$$

Once the degree of $R_r(x) < \left\lfloor \frac{d-1}{2} \right\rfloor$, then $A_r(x) = \Lambda(x)$ and $R_r(x) = \Gamma(x)$.
As in the previous algorithm, an exhaustive method is applied to find the roots of $\Lambda(x)$. These roots correspond to the error locations. The Forney algorithm is computed to calculate the error magnitudes from $\Gamma(x)$. Then the error can be subtracted in the received codeword.
Extended Euclidean algorithm tends to be more used in practice because it is easier to implement, but the Berlekamp–Massey algorithm tends to lead to more efficient hardware and software implementations.

1.4 Product codes

Product codes have been invented in 1954 by Peter Elias in [31]. They are two-dimensional linear codes and in the general case are referred to as multidimensional codes. Product codes are obtained by tensor product of two (or more) linear codes with a structure that is well-suited to iterative decoding via its graphical description.

Given two linear codes C_1 and C_2 of parameters $[n_1, k_1, d_1]_q$ and $[n_2, k_2, d_2]_q$ respectively. Let G_1 and G_2 be two matrices of size $k_1 \times n_1$ and $k_2 \times n_2$, containing in their rows a basis for the subspaces C_1 and C_2 respectively. G_1 and G_2 are called the generator matrices of C_1 and C_2.

Definition 1.10. The product of C_1 and C_2 is the linear code C_p consisting of all $n_1 \times n_2$ matrices with the property that each matrix column is a codeword of C_1 and each matrix row is a codeword of C_2.

From the two generator matrices G_1 and G_2 a product code C_p is constructed as a subspace of F_q^N with a generator matrix $G_p = G_1 \otimes G_2$, where $N = n_1 n_2$ and $\otimes$ denotes the Kronecker product. In concatenated codes terminology, C_1 and C_2 are called component codes.
The parameters of a such product code C_P are defined by:

- Block-length: $N = n_1 n_2$,
- Dimension: $K = k_1 k_2$,
- Minimum Hamming distance: $d = d_1 d_2$,
- Coding rate: $R = \frac{k_1}{n_1} \cdot \frac{k_2}{n_2}$.

More specifications and proofs are available in [10], [52], [61] and [65].
The structure of a systematic $[N, K, d]_q$ product code is illustrated in Figure 1.2.

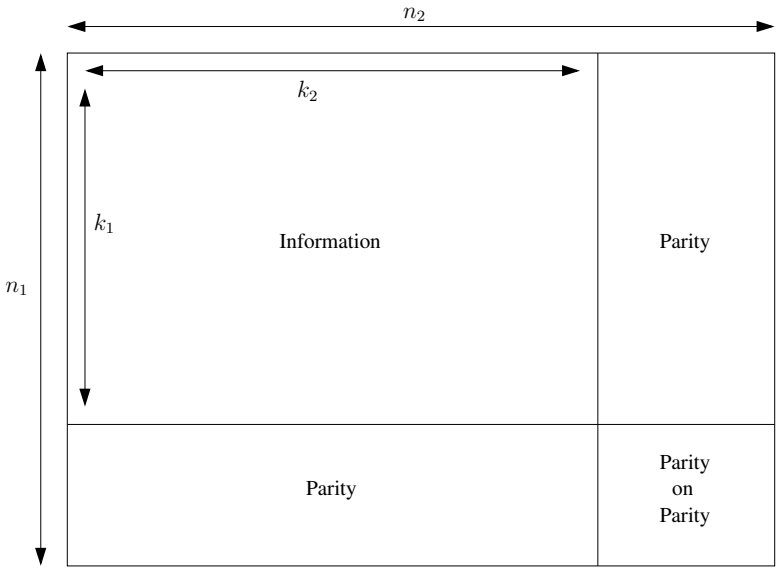


Figure 1.2: Structure of a systematic $[N, K, d_p]_q$ product codeword.

A product code can also be represented by graphs. A complete study of graph representations for product code will be presented later in Chapter 4 Section 4.2.

Product codes received a great attention due to their capability of correcting multiple burst errors [109][117], the availability of erasure-error bounded-distance decoding algorithms [113], their ability of correcting many errors beyond the guaranteed correction capacity [1], and their efficient implementation with a variable rate [115]. The excellent performance of iterative (turbo) decoding of product codes on the Gaussian channel [76] made them compete with Turbo codes and LDPC codes for short and moderate block-length.
The class of product codes in which the row and the column code are both Reed- Solomon codes was extensively used since more than two decades in DVD storage media and in mobile cellular networks [116]. In these systems, the channel is modeled as a symbol-error channel without soft information, i.e. suited to algebraic decoding. Improvements were suggested for these RS-based product codes such as soft information provided by list decoding [90] within the iterative process in a Reddy-Robinson framework [82]. Reed-Solomon based product codes were directly decoded via a Guruswami-Sudan list decoder [43] after being generalized to bivariate polynomials [4]. For general tensor products of codes and interleaved, a recent efficient list decoding algorithm was published [39], with an improved list size in the binary case. On channels with soft information, RS-based product codes may be row-column decoded with soft-decision constituent decoders [32], [48].

In this thesis, product codes are studied in the context of distributed storage. Therefore, channels with independent erasures or block multiple erasures are considered. Four decoding methods are known for product codes over erasure channels:

- ML decoder. This is a non-iterative decoder. It is based on a Gaussian reduction of the parity-check matrix of the product code.

- Iterative algebraic decoder. At odd decoding iterations, component codes C_1 on each column are decoded via an algebraic decoder (bounded-distance) that fills up to $d-1$ erasures. At even decoding iterations, component codes C_2 on each row are decoded via a similar algebraic decoder. An example of algebraic decoder for Reed-Solomon component codes is the Berlekamp-Massey decoder presented in Section 1.3.

- Iterative ML-per-component decoder. This decoder was considered by Rosnes in [88] for binary product codes. At odd decoding iterations, column codes C_1 are decoded via an optimal decoder (ML for C_1). At even decoding iterations, row codes C_2 via a similar optimal decoder (ML for C_2).

- Iterative belief-propagation decoder based on the Tanner graph of $C_{\mathcal{P}}$, as studied by Schwartz et al. for general linear block codes [93] and by Di et al. for low-density parity-check codes [27].

1.5 Low-density parity-check codes

Low-Density Parity-Check (LDPC) codes are linear error-correcting codes based on parity-check matrices. LDPC codes were developed by Robert G. Gallager in his doctoral dissertation in 1963 [36], hence so-called Gallager codes. In [101], Tanner proposed a bipartite graph representation of LDPC codes. Moreover, LDPC codes belong to the family of sparse graph codes, which are not MDS codes. However, some existing practical constructions of sparse graph codes allow to approach very closely the Shannon limit over a symmetric memoryless channel, in particular over the BEC. Thus, they are named capacity-approaching codes. Using iterative belief propagation techniques, LDPC codes can be decoded in time linear to their block-length. All these advantages explain that LDPC codes are widely used in applications requiring reliable and highly efficient information transfer over noisy channels.

Graph representation of codes

Any linear code can be represented as a graph. We start by giving the general definition of a graph [11].

Definition 1.11. A graph $\mathcal{G}$ is an ordered pair of disjoint sets (V, E) such that E is a subset of the set of unordered pairs of V. The set V is the set of vertices and the set E is the set of edges.

LDPC codes are represented by a bipartite graph also called Tanner graph [101]. It is defined as follows.

Definition 1.12. A graph $\mathcal{G}$ is a bipartite graph with vertex classes V_1 and V_2 if $V(\mathcal{G}) = V_1 \cup V_2$, $V_1 \cap V_2 = \emptyset$ and every edge joins a vertex of V_1 to a vertex of V_2, i.e. $\mathcal{G}$ has a bipartition (V_1, V_2).

An LDPC code shall be represented by a bipartite graph $\mathcal{G} = (V_1, V_2, E)$. V_1 is the set representing the symbols from the finite field $\mathbb{F}_q$, V_2 is the set representing the constraints to be applied on these symbols, and E is the set of edges. We adopt the following representation:

- Symbols also called variable nodes are represented by circles on the left of the bipartite graph.
- Constraints on symbols also called check nodes are represented by squares on the right of the bipartite graph.

The incident matrix of $\mathcal{G}$ is a matrix that shows the relationship between the two sets of nodes. The matrix has one row for each element of V_2 and one column for each element of V_1. The entry in row i and column j is 1 if i and j are linked by one edge in the graph and 0 if they are not. The incident matrix of $\mathcal{G}$ is a parity-check matrix H of the LDPC code. A check node in the Tanner graph corresponds to a row in H, i.e. it represents a parity-check equation (SPC) code such as $c_1 + c_2 + c_3 = 0$ shown in Figure 1.3 .

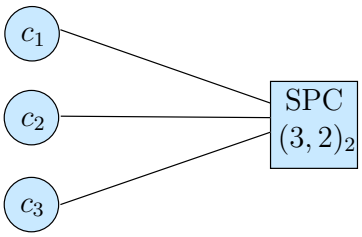


Figure 1.3: Representation of a Single Parity Check $\text{SPC}(3,2)_2$, and constraints on variable nodes. Variable nodes on the left are represented by circles and check nodes on the right are represented by squares.

LDPC codes construction

An LDPC of length N and dimension K is defined by a binary sparse $(N-K) \times N$ parity-check matrix H. The LDPC code is represented by a bipartite graph, depicted in Figure 1.5, with N variable nodes and $N-K$ check nodes. An example of LDPC code protograph and its corresponding compact Tanner graph is given in Figure 1.4. A protograph is a type of compact Tanner graphs.

For regular LDPC codes, all variable nodes have the same degree d_b and all check nodes have the same degree d_c. The number of edges $|E|$ is given by $|E| = d_b N$. The sparsity of the LDPC codes means that left degree and right degree are small compare to the number of edges.

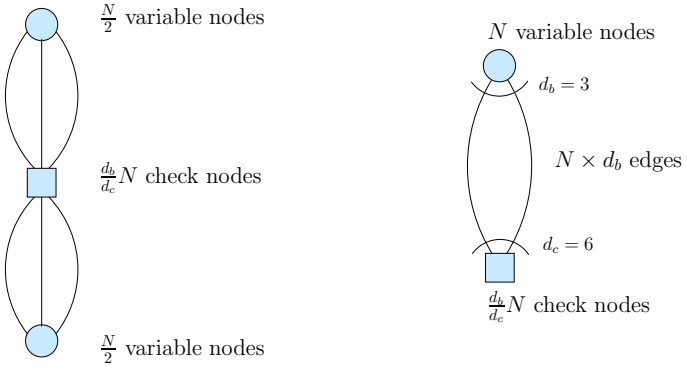


Figure 1.4: Protograph of a $(3,6)$ LDPC code (left) and an equivalent compact Tanner graph (right).

To choose a random graph, consider the sockets on each side of the bipartite graph. A socket is a place where an edge is attached. So, each variable node has d_b sockets and each check node has d_c sockets. A random graph is formed by matching up the variable node sockets and the check node sockets, using a random permutation. One right socket is linked to one left socket by one edge. Therefore, there exists $|E|!$ matching between right and left sockets. A (d_b, d_c) LDPC ensemble is defined as one random permutation among the $|E|!$.

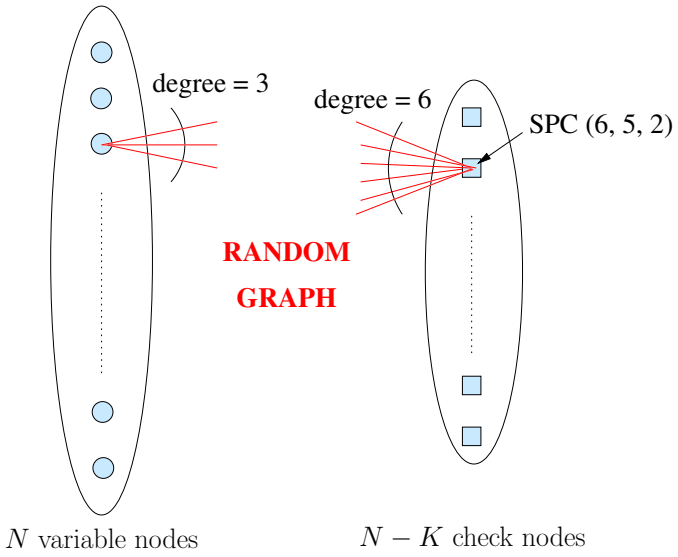


Figure 1.5: Tanner graph representation of a $(d_b = 3, d_c = 6)$ LDPC code. LDPC code has length N, dimension K and coding rate $R = 1/2$. The $3N$ edges are randomly chosen.

The coding rate of a regular (d_b, d_c) LDPC code, assuming that all these check equations are linearly independent, is defined as:

$$R = \frac{K}{N} \geq 1 - \frac{d_b}{d_c}. \tag{1.33}$$

Irregular LDPC codes

The idea of irregular LDPC codes was first introduced by Luby et al. in [62]. An LDPC code is said to be irregular if graph nodes of the same kind do not have equal degree.

Thereby, variable nodes have different degrees. The fraction of edges emanating from variable nodes of degree i is denoted λ_i. The degree distribution is defined by

$$\lambda(x) = \sum_{i=2}^{d_b} \lambda_i x^{i-1}, \tag{1.34}$$

where d_b is now defined as the maximal degree of variable nodes, $\lambda(1) = 1$, and $0 \leq \lambda_i \leq 1$ for all i. Assume the code has N variable nodes. The number N_i of variable nodes of degree i is then

$$N_i = N \frac{\lambda_i / i}{\sum_{j=2}^{d_b} \lambda_j / j}. \tag{1.35}$$

Similarly, check nodes have different degrees. The fraction of edges connected to check

nodes of degree j is ρ_j. The degree distribution is defined by

$$\rho(x) = \sum_{j=2}^{d_c} \rho_j x^{j-1}, \tag{1.36}$$

where d_c is now defined as the maximal degree of check nodes, $\rho(1) = 1$, and $0 \leq \rho_j \leq 1$ for all j. Assume the code has L check nodes. The total number of edges $|E|$ is given by

$$|E| = \frac{N}{\sum_{i=2}^{d_b} \lambda_i / i} = \frac{L}{\sum_{i=2}^{d_c} \rho_i / i}. \tag{1.37}$$

From this equation, the number of check nodes is:

$$L = N \cdot \frac{\sum_{i=2}^{d_c} \rho_i / i}{\sum_{i=2}^{d_b} \lambda_i / i} \tag{1.38}$$

Let $\bar{d}_b$ and $\bar{d}_c$ be the average degree of variable nodes and check nodes respectively. They can be expressed as

$$\bar{d}_b = \frac{|E|}{N} = \frac{1}{\sum_{i=2}^{d_b} \lambda_i / i} \tag{1.39}$$

$$\bar{d}_c = \frac{|E|}{L} = \frac{1}{\sum_{i=2}^{d_c} \rho_i / i} \tag{1.40}$$

The coding rate of an irregular LDPC code, assuming that all these check equations are linearly independent, is equal to

$$R = \frac{K}{N} = 1 - \frac{L}{N} = 1 - \frac{\bar{d}_b}{\bar{d}_c} \tag{1.41}$$

Multi-edge type LDPC codes

Another group of LDPC codes, presented by Tom Richardson in 2002 in [85], is multi-edge type LDPC codes. Multi-edge type LDPC codes are a generalization of regular and irregular LDPC codes. A multi-edge type LDPC ensemble is constituted of a finite number of edge types. Multiple classes of variable nodes and check nodes are created. Different node types may have different received distributions, i.e, the associated bits may go through different channels. Multiple-edge type LDPC structures are proposed in [16] and [17]. In Chapter 3.3.1, the properties of Root-LDPC codes, which are multi-edge type LDPC codes, are discussed and detailed.

Decoding LDPC codes

The decoding process of an LDPC code is based on its graph representation. Operations are made on variable nodes and check nodes.
In presence of a uniform source encoded by an $[N, K]$ LDPC code whose codewords are transmitted on an i.i.d. binary erasure channel, the iterative non-probabilistic decoding algorithm is given by the following steps:

1. Initialize $Iter = 0$ and $j = 1$.
2. Count the number μ of erased bits connected to check node j.
3. If $\mu = 1$ then fill the erased bit by summing other bits modulo 2.
4. Increment j. If $j > N - K$ then increment $Iter$ and set $j = 1$.
5. if $Iter > Maxiter$ then Stop else Goto Step 2.

Now let us analyze the asymptotic performance of the iterative decoding of LDPC codes over the BEC. This asymptotic analysis is made via density evolution (DE). Assume an infinite length code with a graph representation without cycles. Let x_i denote the erasure probability at iteration i. By looking to the tree neighborhood of a graph node, it is easy to show that erasure probability x_i, after LDPC decoding at iteration i, on the binary erasure channel satisfies:

$$x_{i+1} = \epsilon\lambda(1 - \rho(1 - x_i)), \tag{1.42}$$

where ϵ is the channel erasure probability.

Theorem 1.1. *Let $f(\epsilon, x) = \epsilon\lambda(1 - \rho(1 - x))$ where $0 \leq \epsilon \leq 1$. There exists a threshold w^{BP} such that*

1. *ϵ^{BP}=sup$\{\epsilon \in [0, 1] : x = f(\epsilon, x)$ has no solution in $[0, 1]\}$*
2. *ϵ^{BP}=inf$\{\epsilon \in [0, 1] : x = f(\epsilon, x)$ has a solution in $[0, 1]\}$*

For a degree distribution (λ, ρ) with a BP threshold ϵ^{BP}, x^{BP} is a critical point iff

$$f(\epsilon^{BP}, x^{BP}) = x^{BP} \quad \text{and} \quad \frac{\partial f(\epsilon^{BP}, x)}{\partial x} \mid_{x=x^{BP}} = 1.$$

Now let us investigate the stability condition. The stability condition gives an upper bound on the BP threshold. The stability condition of the fixed-point at the origin for a function $f(x)$ is obtained by writing that $f'(0) \leq 1$. Thus, the necessary and sufficient condition for stability of 0 is

$$\epsilon^{BP} \leq \frac{1}{\lambda'(0)\rho'(1)}. \tag{1.43}$$

This bound is also true for a Maximum-A-Posteriori (MAP) decoding. Another method to visualize the asymptotic performance under BP decoding is Extrinsic Information Transfer (EXIT) chart. The definition of EXIT function is given below.

Definition 1.13. Given an $[N, K]$ linear code of rate $R = \frac{K}{N}$ and X chosen with uniform probability from $[N, K]$, let Y denote the result of letting X be transmitted over a BEC(ϵ). The EXIT function $h_i(\epsilon)$ is defined as [86]

$$h_i(\epsilon) = H(X_i \mid Y_{-i}). \tag{1.44}$$

The average EXIT function is

$$h(\epsilon) = \frac{1}{N}\sum_{i=1}^{N} h_i(\epsilon). \tag{1.45}$$

The most relevant property of EXIT function is the Area theorem which can be expressed as follows,

Theorem 1.2. *Let $[N, K]$ be a linear code, then*

$$\int_0^{\epsilon} h(x)dx = \frac{1}{N}H(X \mid Y), \tag{1.46}$$

where $H(X \mid Y)$ is the conditional entropy of the codeword X given the observation Y at the receiver.

$$\int_0^{1} h(x)d(x) = R = \frac{K}{N}. \tag{1.47}$$

BP EXIT function for a (d_b, d_c) regular LDPC ensemble on the BEC(ϵ) is [86]

$$h^{BP}(\epsilon) = \begin{cases} (\epsilon,\ 0), & \epsilon \in [0, \epsilon^{BP}[\\ \left(\frac{x}{(1-(1-x^{d_c-1}))^{d_b-1}},\ (1-(1-x^{d_c-1}))^{d_b}\right), & \epsilon \in]\epsilon^{BP}, 1[. \end{cases} \tag{1.48}$$

Therefore, we can visualize non-trivial fixed-point by plotting the following pairs:

$$\left(\frac{x}{(1-(1-x^{d_c-1}))^{d_b-1}},\ (1-(1-x^{d_c-1}))^{d_b}\right), \tag{1.49}$$

in the unit box $[0, 1] \times [0, 1]$. Beyond the BP threshold, two non trivial fixed-points can be observed (stable and unstable).

In a similar way, let us defined h^A, $h^{MAP}$and h^S, the entropy-wise channel parameters for a (λ, ρ) LDPC code:

Definition 1.14. Consider a code with a coding rate $R = 1 - \frac{d_b}{d_c}$, the Shannon threshold is

$$h^S = \frac{d_b}{d_c}. \tag{1.50}$$

Definition 1.15. Under MAP decoding and after a large number of iterations, the bit error probability of the ensemble is vanishing for $h < h^{MAP}$, where h is the channel parameter.

Definition 1.16. The area under the EXIT curve in the range $[h^A, 1]$ is equal to the ensemble coding rate.

We can notice that:

$$h^{BP} \leq h^{MAP} \leq h^{A} \leq h^{S}. \tag{1.51}$$

On the binary erasure channel, the area threshold h^A and the MAP threshold h^{MAP} are equal.

For decoding LDPC codes over binary symmetric channels, a well known algorithm is the sum-product decoding. This decoding technique is sub-optimal. However it is based on iterative belief propagation decoding and gives excellent results when practically implemented.
The sum-product decoding techniques view each parity check as an independent SPC code. Each SPC code is separately decoded using soft-in soft-out (SISO) decoder which is capable of determining the *a-posteriori* probability. The soft decision information from each SISO decoding is cross-checked and updated with other redundant SPC decodings of the same information bit. Each SPC code is then decoded one more time using the updated soft decision information. This process is iterated until a valid codeword is obtained or decoding is exhausted.

1.6 Fountain codes

Fountain codes are a class of erasure codes which can generate a potentially limitless sequence of encoded symbols from a given set of k source symbols. Therefore, Fountain codes are rateless and can be encoded on the fly. At the decoder side, the original source symbols can be ideally recovered from any subset of the encoded symbols. The size k' of this subset is equal to or only slightly larger than the number of source symbols, i.e. $k' \geq (1+\kappa)k$ with κ arbitrarily small. A Fountain code is optimal if the original k source symbols can be recovered from any k' encoded symbols. Contrary to LDPC codes, the Fountain codes construction ensures the sparsity of the generator matrix.

The first practical realization of Fountain codes was proposed in 1998 by Michael Luby in [63]. Hence, these codes were called Luby Transform (LT) codes . These near-optimal erasure codes suffer from encoding and decoding complexities that grow as $O(\log(\frac{k}{\delta}))$ per symbol, where δ is the target error probability. In order to ensure efficient LT encoders and decoders, Raptor codes were subsequently introduced by Shokrollahi in [96]. They achieve linear time encoding and decoding complexity as $O(\log(\frac{1}{\kappa}))$ (per symbol) through a precoding stage of the input symbols.

Fountain codes encoder

The encoding process of Fountain codes is defined by a sparse bipartite graph connecting coded symbols to information symbols. Symbols shall also be called nodes in this section. For each encoded node, we associate a degree d, randomly chosen from a specific degree distribution $\rho(d)$ defining the Fountain code, the reader should refer to chapter 50 in [64]. The degree d of an encoded symbol is the number of its connections to data symbols. For LT codes, the degree of each encoded nodes is uniformly drawn from robust Soliton

distribution[63]. The degree of data symbols may be left without any control (Poisson distributed). The edges between encoded nodes and input nodes are drawn following an independent and identically distributed random variable.

Fountain codes are fully determined by a sparse $k \times n$ generator matrix G. A column j of the generator matrix corresponds to the construction of one encoded node and has some Hamming weight d_j. For binary codes, if the jth encoded node is linked with the ith source node, then a 1 is in position i in the jth column, otherwise it is a 0. A codeword $\mathbf{v}_{1\times n}$ can be written as: $\mathbf{v} = \mathbf{u}G$ where $\mathbf{u}_{1\times k}$ is an input vector. The original source data is divided into k chunks. These k symbols correspond to the left k nodes of the bipartite graph illustrated on Figure 1.6. Fountain codes are commonly used with a limitless sequence of encoded symbols.The n output nodes are drawn on the right of the bipartite graph. Contrary to LDPC codes, there are variable nodes on both sides of the bipartite graph.

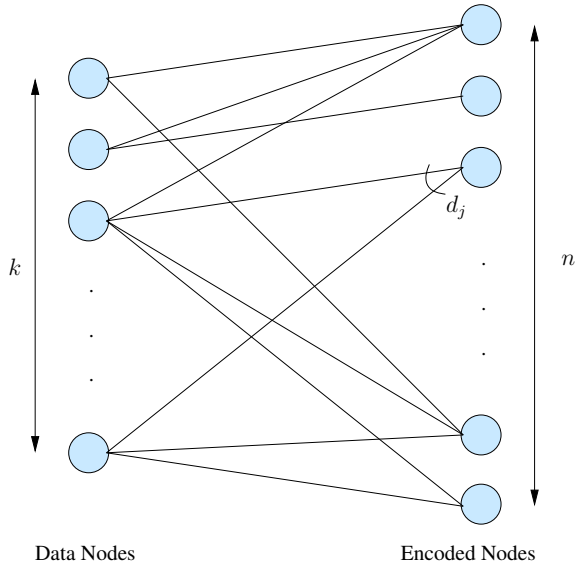


Figure 1.6: Example of bipartite graph of Fountain code.

Figure 1.6 illustrates an example of Fountain codes, each output node is created by XORing the input nodes connected to it. Input nodes connected to one output node are called neighbors of this node.

The encoding process of one output node can be summarized as following:

1. Randomly choose the degree d_j of the encoding jth symbol from a degree distribution $\rho(d)$.

2. Choose, uniformly at random, d_j distinct input symbols as neighbors of the encoding symbols.

3. Encode the output symbol by XORing the d_j data symbols.

Fountain codes decoder

Fountain's decoder is an iterative process using message-passing algorithm over a binary erasure channel. Figure 1.7 illustrates a simple example of Fountain code decoding with $k = 3$ and $n = 6$.

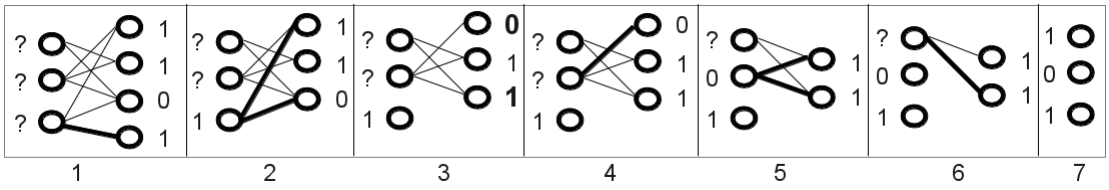


Figure 1.7: Example of Fountain code decoding with $k = 3$ and $n = 4$.

In the first step of Fountain decoding, we look for one output node of degree one. In Figure 1.7, the only node of degree one is the last output node. Then, in a second step, we take out this output node from its neighboring input node after propagating the symbol value from right to left. Now, as the information node is discovered, its value is propagated from left to right to all its neighboring output nodes. This is shown in step 3 in Figure 1.7. This input node is finally disconnected from the graph. These operations are repeated by looking again for a new output node of degree one. The decoding process stops when all input nodes are discovered or when there are no more output nodes of degree one. The decoding process fails if it stops while some of the data nodes are not determined yet.
The steps of the decoding process are summarized below:

1. Find an encoded symbol with degree 1.
2. Set its appropriate data symbol.
3. Extract data symbol from all its neighbors.
4. Repeat until all data nodes are determined.

1.7 Spatially coupled codes

Convolutional LDPC codes were first introduced by Felstrom and Zigangirov [34]. Exceptional thresholds of LDPC convolutional codes [23][60] led to the discovery of threshold saturation [55] and to the general concept of spatial coupling by Kudekar et al. in 2013 [56]. Spatial coupling is currently applied in many areas in coding and information theory. A spatially-coupled LDPC ensemble is formed by multiple LDPC ensembles sharing common edges. For asymptotic coding schemes, the advantages of spatially-coupled ensembles can be recapitulated in three main points: regular degree distributions substituting for complex irregular distributions, iterative belief propagation decoding achieving

maximum-a-posteriori performance, and a universal spatial coupling concept that can be applied in many areas beyond LDPC coding.

Consider a regular (d_b, d_c) LDPC ensemble. The design coding rate is $R = 1 - \frac{d_b}{d_c}$, assume R is fixed. A code in the ensemble has N variable nodes and $\frac{d_b}{d_c}N$ check nodes. Let N be very large ($N \to +\infty$). Call one code in this ensemble as one copy. Imagine that all variable nodes and all check nodes of a copy are located on a vertical line. Put a copy on each position i, $i = -L_c, \cdots, +L_c$. There is no interaction between the copies , not yet. We have a total of $(2L_c + 1)$ spatial positions. The construction of the coupled (d_b, d_c, L_c, w) ensemble is illustrated in Figure 1.8.

Construction of the coupled (d_b, d_c, L_c, w) ensemble:

- The variable nodes are at positions $[-L_c, L_c]$, $L_c \in \mathbb{N}$. At each position there are N variable nodes, $N \in \mathbb{N}$, $N \to +\infty$.
- Check nodes are located at all integer positions from $[-\infty, +\infty]$. There are $\frac{d_b}{d_c}N$ check nodes at each position.
- For a given variable node at position i, its d_b connections are chosen uniformly and independently from check nodes in the range $[i, i + w - 1]$. The positive integer w is a smoothing parameter. In convolutional codes terminology, it is equivalent to the code memory.
- The d_c connections of a check node at position i are uniformly and independently chosen from variable nodes in the range $[i - w + 1, i]$.
- Extra check nodes are added on both size to terminate the coupled chain. Now the coupled chain is terminated on both left and right ends.

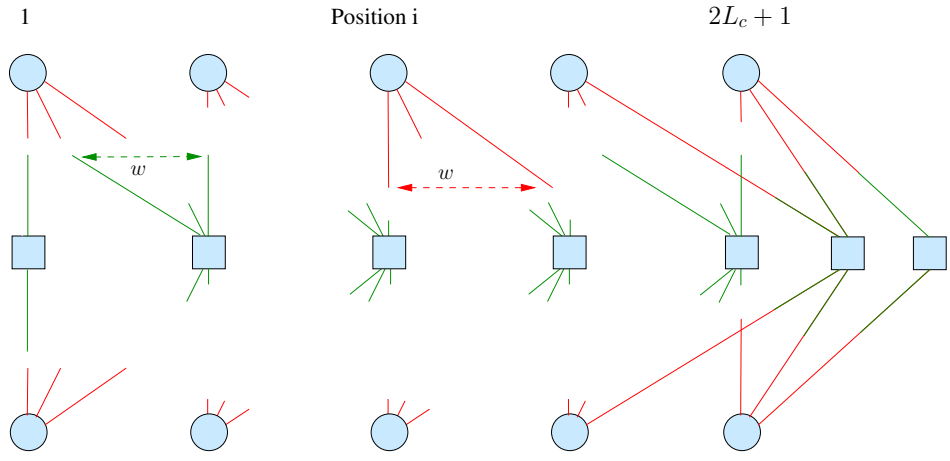


Figure 1.8: Tanner graph of uniform spatially-coupled $(3, 6, L, w)$ ensemble chain.

The ultimate goal of spatial coupling is for a given design coding rate R, on any BMS channel, is to have [55]

$$\lim_{d_b\to\infty}\lim_{w\to\infty}\lim_{L\to\infty} h^{BP} = h^S. \tag{1.52}$$

This goal is achieved into two steps (refer to Equation (1.51)) [55]:

- Step 1- Spatial coupling, $h^{BP}(d_b, d_c, L_c, w) \to h^A(d_b, d_c)$.
- Step 2- Increasing the node degree, $h^A(d_b, d_c) \to h^S$.

Asymptotic analysis of a (d_b, d_c, L_c, w) coupled LDPC ensemble over a BEC(ϵ) is made via density evolution. Density evolution for a uniformly coupled ensemble is expressed as

$$p_i = \epsilon\left(1 - \frac{1}{w}\sum_{j=0}^{w-1}\left(1 - \frac{1}{w}\sum_{k=0}^{w-1} p_{i+j-k}\right)^{d_c-1}\right)^{d_b-1}, \tag{1.53}$$

where p_i is the probability of erasure on BEC at spatial position i, $i = 1 \cdots L_c$.

A new method for spatial coupling of LDPC ensembles is proposed in Chapter 3.1. The method is inspired from overlapped layered coding. Edges of local ensembles and those defining the spatial coupling are separately built. Moreover, we restrict our study to spatial coupling with a short memory $w = 2$.

1.8 Polar codes

A polar code is a linear block error-correcting code invented by Erdal Arikan in 2008 [5]. It is the first code with an explicit construction to demonstrably achieve the channel capacity for symmetric binary-input, discrete, memoryless channels with polynomial dependence on the gap to capacity. Before polar codes, no capacity achieving coding schemes are known for general channels. Thereafter, spatially-coupled codes appear as an alternative to polar codes. Polar codes concept is based on channel polarization. Channel polarization is a method to convert any binary input channel into multiple extremal channels. The extremal channels are:

- Perfect Channel: the output Y perfectly determines the input X, i.e. the channel capacity is one.
- Worst (useless) channel: the output Y is independent of the input X, i.e. the channel capacity is zero.

Channel polarization is information lossless. Encoding and decoding polar codes is low complexity, both are $O(n\log(n))$ which makes them practical for many applications. Assume a binary input alphabet $\mathbb{F}_2$. Consider two copies of a BMS channel $W : \mathbb{F}_2 \to \mathcal{Y}$. The channel splitting is done as follows and is illustrated in Figure 1.9:

$$X_1 = U_1 + U_2, \quad X_2 = U_2, \tag{1.54}$$

where information bits U_1 and U_2 are i.i.d. and uniform on $\mathbb{F}_2$. This transformation corresponds to the 2×2 kernel given by the matrix G_2:

$$G_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \tag{1.55}$$

Channel splitting via G_2 yields two new channels:

$$W^- : \mathbb{F}_2 \to \mathcal{Y}^2 \quad \text{and} \quad W^+ : \mathbb{F}_2 \to \mathcal{Y}^2 \times \mathbb{F}_2. \tag{1.56}$$

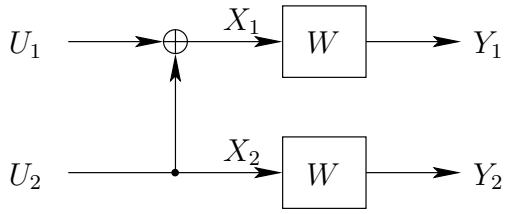


Figure 1.9: Channel splitting into two channels.

Assume W is a BEC(ϵ). Let us introduce W^- which has input U_1 and output (Y_1, Y_2). W^- is equivalent to a BEC(ϵ_1). U_1 is determined if and only if both outputs Y_1 and Y_2 are not erased. So $(1 - \epsilon_1) = (1 - \epsilon)^2$, i.e.

$$W^- = \text{BEC}(2\epsilon - \epsilon^2). \tag{1.57}$$

Similarly, W^+ is a channel which has input U_2 and output (Y_1, Y_2, U_1). W^+ is equivalent to a BEC(ϵ_2). U_2 is erased if and only if both outputs Y_1 and Y_2 are erased. Thus, $\epsilon_2 = \epsilon^2$, i.e.

$$W^+ = \text{BEC}(\epsilon^2). \tag{1.58}$$

What are the properties of the split $W \to (W^-, W^+)$? Let us study the average mutual information. Let $I(W) = I(X;Y)$ where the input distribution is uniform. The quantity $I(W)$ is known as the symmetric capacity of the channel W. Mutual informations of both bad channel W^- and good channel W^+ can be expressed as

$$I(W^-) = I(U_1; Y_1, Y_2) \quad \text{and} \quad I(W^+) = I(U_2; Y_1, Y_2, U_1). \tag{1.59}$$

An obvious property is

$$I(W^-) \leq I(W) \leq I(W^+). \tag{1.60}$$

From the independence of U_1 and U_2, and the chain rule we can deduce that

$$\frac{1}{2} I(W^-) + \frac{1}{2} I(W^+) = I(W). \tag{1.61}$$

We can conclude from this equation that the mutual information is preserved via the polar transform. Now, we provide a simple example. Let W be a BEC(ϵ)

$$\Delta_{BEC}(I) = I(W^+) - I(W^-) = 2(I(W) - I^2(W)), \tag{1.62}$$

where $I(W) = 1 - \epsilon$ for $0 \leq \epsilon \leq 1$.
In the same way, for the BSC(p),

$$\Delta_{BSC}(I) = I(W^+) - I(W^-) = 2[H_2(2p(1-p)) - H_2(p)], \tag{1.63}$$

where $H_2(p)$ is the binary entropy function, for $0 \leq p \leq \frac{1}{2}$.
For any BMS channel, by information combining arguments, we have

$$\Delta_{BSC}(I) \leq \Delta_{BMS}(I) \leq \Delta_{BEC}(I). \tag{1.64}$$

Now let us repeat the splitting l times. For a 2×2 kernel, the transformation is given by $G_2^{\otimes l}$, where $\otimes$ is the Kronecker product. The channel is splitted into $n = 2^l$ channels. The mutual information of channel number i is $I(U_i; Y_1^n, U_1^{i-1})$, for $i = 1, \ldots, n$.

Definition 1.17. The channel polarization as stated in [5] refers to the phenomenon defined by

$$\lim_{n\to\infty} \frac{1}{n} \#\{i : I(U_i; Y_1^n, U_1^{i-1}) \in]\epsilon, 1-\epsilon[\} \to 0, \tag{1.65}$$

i.e., almost all channels are extremal when $n = 2^l$ gets large.

When polarization happens, good channels are used to transmit uncoded bits. The input to bad channels can be *frozen* to a fixed value. Since polarization is information lossless, the data rate which is equal to the fraction of good channels must be $I(W)$.

The decoding process is a successive decoding. The decoder can decode U_1, $U_2, \ldots, U_n$ successively. This decoder is also called successive cancellation. Alternative to successive decoding are:

- Belief propagation decoding of a polar code [50]. Roughly, improvement of error rate by a factor of 10 on all channels.
- List decoding of a polar code [100]. Gain in signal-to-noise ratio is about 0.5 dB.

1.9 Reed-Muller codes

Reed–Muller (RM) codes are a family of linear error-correcting codes. They are named after Irving Reed and David Muller. Muller discovered the codes [68], and Reed proposed the majority logic decoding[83]. They are among the oldest and simplest codes to construct. Reed-Muller codes are extended cyclic codes. RM codes can be considered as a special case of Polar codes where bits are frozen in order to maximize the minimum Hamming distance. More recently, it has been proved that RM codes achieve capacity on the BEC under MAP decoding [58] [57].

Let us give a brief description of Reed-Muller codes. For any m and any r, $0 \leq r \leq m$, there is a binary rth order RM code $R(r, m)$ with the following properties: [65]

- Block-length: $n = 2^m$,
- Dimension: $k = \sum_{l=0}^{r} \binom{m}{l}$,
- Minimum Hamming distance: $d = 2^{m-r}$.

RM codes are related to binary functions on the field $\mathbb{F}_{2^m}$ over the elements $\{0, 1\}$. $R(r, m)$ consists of all polynomials in the binary variable $v_1, \cdots, v_m$ of degree equal to, or less than r. Consider an $R(r, m)$ Reed-Muller code, its dual is the RM code of parameters $R(m - r - 1, m)$ for $0 \leq r \leq m - 1$. Another property of RM codes is that a Reed-Muller codes of length 2^{m+1} can be easily constructed from Reed-Muller codes of length 2^m using the $(u \mid u + v)$ construction:

$$R(r + 1, m + 1) = \{(u \mid u + v) : u \in R(r + 1, m), v \in R(r, m)\} \tag{1.66}$$

The generator matrix of a $R(r, m)$ code is denoted $G(r, m)$. The equivalent statement of Equation (1.66) is:

$$G(r, m) = \begin{bmatrix} \text{G(r+1,m)} & \text{G(r+1,m)} \\ 0 & \text{G(r,m)} \end{bmatrix} \tag{1.67}$$

Some special cases of Reed–Muller codes are listed below:

- $R(0, m)$ code is the repetition code of block-length $n = 2^m$.
- $R(m, m)$ code is the universal code of block-length $n = 2^m$, i.e. the set of all binary words of length 2^m.
- $R(m - 1; m)$ code is the parity check code of length $n = 2^m$.

Another advantage of RM codes is the ease with which they can be decoded via majority logic decoder. This technique was proposed by Irving Reed [83]. A majority logic decoding is a method to decode repetition codes, based on the assumption that the largest number of occurrences of a symbol was the transmitted symbol. Applied to RM decoding, the key idea of majority logic decoder is to build several check-sums for each received symbol. All the different check-sums for one received symbol must have the same value: the weight of the information symbol. Consequently the majority logic decoder can be used to decode the value of the received symbol. Once each order of the polynomial is decoded, the received symbol is modified accordingly by removing the corresponding check-sums, up to the present iteration. For an $R(r, m)$ code, the decoding is iterated $r + 1$ times before reaching the final received symbol. This method allows to calculate the values of information bits. Then, the received symbol can be reconstructed by multiplying the information symbol with the generator matrix. If the symbol is successfully decoded via majority logic decoding, at the end of the $r + 1$ decoding iterations, the modified received symbol should be all-zero. This decoding method can be applied to other finite geometry codes. More recently, list decoding of Reed-Muller codes were also proposed [72][38].

1.10 Conclusions

This chapter is a brief overview of erasure codes. Coding theory is in constant development. Codes have to be designed for different channel models. In this thesis, we focus on erasure channels. The erasure channel is currently a major area of research in coding theory [57][58] because of strong connections with theoretical computer science and its mathematical simplicity that easily allow to understand the behavior of codes such as LDPC codes [27] and general linear block codes [93]. In the past three years there has been a lot of innovations in the field of modern coding theory [86][89]. In parallel to coding theory, a part of information theory is focusing on finite length regime where information rate and code performance are studied at finite block length instead of asymptotic length.

Coding theory has applications in many areas such as data compression, cryptography, classic error-correction for noisy channels, network coding, and more recently distributed storage. In the next chapter, an overview of codes for distributed storage is made. Constraints for distributed coding are slightly different than classic coding theory. They are studied and discussed in the sequel.

Chapter 2

Coding for Distributed Storage

Error-control codes are one of the main tools for distributive computing/storage. Besides trivial repetition coding, a Reed-Solomon code or any MDS code is capable of reconstructing the source file [54]. More recently, new coding schemes specifically designed for distributed storage applications have been developed [29][25][70]. We aim in this chapter to present distributed storage constraints and overview existing codes.

2.1 Distributed storage parameters and constraints

In this thesis, only storage networks with a large number of servers are considered. Before defining distributed storage systems parameters and issues, let us take a look at the description of Facebook warehouse cluster organization. This practical case will allow us to better understand and target the stakes of distributed storage. The whole description is available in [80]. A warehouse cluster is divided into two Hadoop Distributed File System (HDFS) clusters. HDFS is an open-source software framework written in Java for distributed storage and distributed processing of very large data sets on computer clusters built from commodity hardware. These HDFS clusters store hundreds of petabytes of data. Each cluster contains a few thousand of machines. Each machine stores up to 36 terabytes of data. The most frequently accessed data is stored as three replicas. Data which has not been accessed for more than three months are coded with a $[14, 10]$ Reed-Solomon code. The different blocks of a codeword are placed on different randomly-chosen machines. In the following a machine can be referred to as a node of the storage network. A median of 50 machines are unavailable for more than 15 minutes in a day in the warehouse cluster. When blocks are missing in a codeword, on average 98.08% have exactly one block missing, 1.87% have two blocks missing and only 0.05% have three or more blocks missing.
From the study of Facebook warehouse cluster, we can abstract the five main points of the design of good codes for distributed storage systems.

- Encoding. This constraint is directly linked to storage space utilization in data

centers. The service sold by the cloud provider to an end-user has a level of reliability. One of the main challenges for cloud providers is to make a tradeoff between a level of reliability proposed to the client and the storage space dedicated to the data. Indeed, the level of reliability depends on the amount of redundancy in the code structure while the storage space in a warehouse cluster has to be optimized. Therefore, the choice of the code parameters defining the coding rate is crucial. In addition, systematic encoders, referred to in Definition 1.9, are more suitable for distributed storage since the time needed to read the information data is minimized.

- Allocation. The allocation problem can be stated in one question. How to distribute the coded symbols across the network in order to be robust against failures?
- Updating. The code structure has to avoid a complete re-encoding of data when updating them. Some applications do not need to update the data once stored.
- Reconstruction. In a distributed storage system when nodes failures occur, the main objective is to reconstruct the failed nodes such that the client/collector is always able of recovering its data by contacting any subset of the storing nodes. Two different methods exist to regenerate the failed node. When the failed node is exactly regenerated by restoring the lost encoded data with their exact replicas the process is called exact repair. However, this requirement can be relaxed and functional repair is another way to regenerate a node. The newly generated blocks may contain data which is different from that of the failed node as long as the repaired system maintains the code properties. From Facebook warehouse cluster statistics, one node failure is the most common scenario.
- Data downloading. For distributed storage applications, the choice of the decoding method is yet another critical challenge. Code design should allow for an iterative decoding process in order to avoid the high complexity of Maximum Likelihood (ML) decoding. When using an ML decoder, all nodes have to be contacted in order to repair the failed node resulting in a high decoding complexity and a high consumption of network bandwidth.

The main objectives of coding for storage are to increase the reliability of data reconstruction, while minimizing the required repair bandwidth and the number of nodes to be contacted per repair. Indeed, the general network traffic should not be hogged by the cross-rack traffic due to the recovery operations. Codes dealing with this constraint are called regenerating codes [28].

At the same time, during a repair process, the number of nodes to be contacted per repair has to be reduced, i.e. code locality has to be low. The concept of code locality was first defined by Gopalan et al. as the locality of information [40]. The code locality is given by Definition 2.1 in the standard context of independent failures (independent erasures). The locality of a symbol is the number of symbols that need to be accessed in order to reconstruct this symbol. Fast reparation is ensured by a low locality of symbols. A new class of codes minimizing locality appeared in the literature. These codes

are called Locally Repairable Codes and were introduced in [49] and developed in, e.g. [119],[3], and [70].

Definition 2.1. Consider a $C[n, k, d]_q$ linear code over a finite field $\mathbb{F}_q$. A codeword in C is written as a vector of n symbols $(c_1, c_2, \ldots, c_n)$. Information locality, $Loc(C)$, of C is defined as:

$$Loc(C) = max_i\ Loc(c_i),$$

where a code symbol c_i has locality $Loc(c_i) = r$ if it can be recovered from accessing only r other code symbols, i.e. $c_i = \sum_{j=1}^{r} \lambda_j c_j$, where $\lambda_j \in F_q$.

All the constraints of coding for distributive storage have been presented. The different coding schemes implemented in today's data centers are now reviewed.

2.2 Today's methods: Replication and erasure codes

Nowadays, coding for distributed storage systems is achieved using two main approaches: Replication and Erasure codes. Replication, which is the most widely used method, consists in storing a copy of the entire data on several storage nodes. In data centers for instance, users' data is replicated on three different storage nodes. This allows to restore data easily in case of node failure. Indeed, the data is recovered from one of the other available nodes, and stored on a new healthy node. From error-correcting coding perspective, replication is a repetition code. It is simple to implement but is inefficient in terms of storage space utilization.

In order to overcome this problem, an MDS erasure codes approach is used in some storage systems such as Facebook. In this case, initial data is divided into k chunks and redundancy is added to create n blocks which are distributed across n nodes on the network. Thus, each node stores only a part of the encoded data. This allows an end user to recover the entire data by downloading coded chunks from any k or more neighboring nodes. When a storage node fails, it is replaced by a new empty node which downloads the entire data from its k neighbors to recover the data stored prior to failure. Consequently, at least k nodes have to be reachable at any time.
Erasure codes are inefficient because the replacement node is required to download the whole data of k coded nodes and then reconstruct the entire data, although it eventually stores only a small part of it. They suffer from a very slow recovery process, in particular, a slow repairing of a single node failure, which is the common failure scenario. Nevertheless, erasure codes provide a better reliability-redundancy tradeoff than repetition codes. Hence, an important task in the design of erasure codes is to accomplish fast recovery and yet support a large number of node failures.

Despite their ill-suited characteristics for distributed storage constraints, MDS erasure codes, namely Reed-Solomon codes, are widely used. The main reason is historical: When the first data storage networks appeared, the research concerning the other erasure-correcting codes was not so advanced. Moreover, RS codes have been implemented in RAID systems and for a long time, the different RAID levels met the needs of users. With the emergence of big storage networks, i.e. with a large number of servers/nodes, new coding schemes based on RS modifications have been developed [91][78][33]. These codes are designed in order to enhance RS code locality and overcome the repair process of MDS erasure codes.

Recently, in the literature, other coding techniques have been proposed to overcome the repair process drawbacks of MDS erasure codes, namely regenerating codes and locally repairable codes. They are described in the next sections.

2.3 Regenerating codes

Regenerating codes were introduced by Dimakis et al. in [28] as a new class of exact repair codes. In a way similar to erasure codes, each node stores small chunks of the encoded data and the end user can recover the original data from any k nodes. However, in regenerating codes, upon failure of a node, the replacement node is able to download a portion of the data stored in any $d \geq k$ nodes in order to recover the data prior to failure, and not the entire data as in erasure codes [79][80][29]. Thus, they allow a drastic reduction in the amount of data downloaded for repair.

In the current literature, two classes of regenerating codes have emerged, mainly the class of minimum bandwidth regenerating (MBR) codes and the class of minimum storage regenerating (MSR) codes. MSR codes are optimal storage efficiency codes. They require a minimal storage space per node. These codes attain the MSR point of the optimal storage-bandwidth tradeoff curve. The MSR point is an extremal point of this curve as described in Figure 2.1. MBR codes are optimal bandwidth efficiency codes. They require minimum repair-bandwidth, i.e. a replacement node downloads precisely the data that it wants to store. These codes attain the MBR point of the storage-bandwidth tradeoff curve. The MBR point is the second extremal point of the optimal storage-bandwidth tradeoff illustrated in Figure 2.1. This tradeoff is determined using a network coding inspired analysis, assuming that each new coming node contacts at least k live nodes for each repair.

In Figure 2.1, α is the amount of storage space per node, γ is the amount of bandwidth needed to regenerate a lost node and t represents the multiple repairs carried out simultaneously. Node repair is required to be accomplished by connecting to any d nodes and downloading $\beta \leq \alpha$ symbols from each node. The repair bandwidth is defined by the product $d \times \beta$. In [79], an explicit construction of MBR codes is proposed for all feasible values of parameters $[n, k, d]$ and MSR codes for all $[n, k, d \geq 2k - 2]$, using a product-matrix framework. Explicit codes of parameters $[n = d + 1, k, d]$ are also pre-

sented in [79]. These codes involve copying of data without any computation, they are uncoded-repair codes.

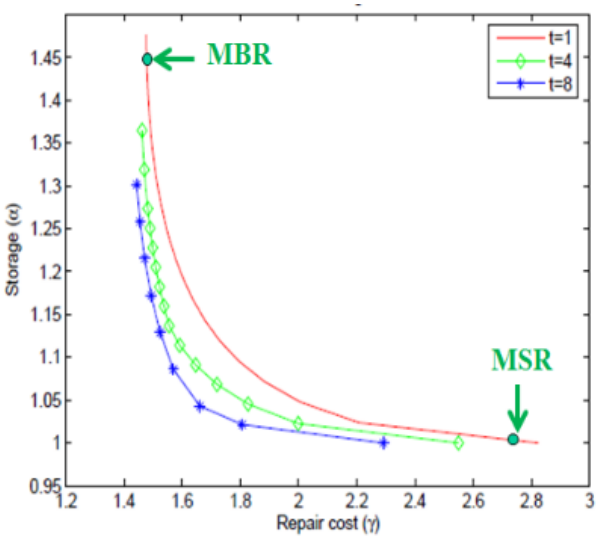


Figure 2.1: Storage-Bandwidth tradeoff [25].

2.4 Locally repairable codes

Locally repairable codes are another coding approach that exploits the locality concept of codes. This concept was first defined by Gopalan et *al.* in [40], as the locality of information. It led to the introduction of codes with locality. These codes are designed such that the number d of nodes to be contacted for repairing a failed node is smaller than k, $d \leq k$. Gopalan et al. proposed in [40] a modified version of the Singleton bond for locally reparable codes. Consider a linear code $C[n, k, d]_q$ with locality $Loc(C)$, the relation between the code locality and its parameters is given by

$$d \leq n - k - \lceil \frac{k}{Loc(C)} \rceil + 2. \tag{2.1}$$

A new class of codes with locality, presented by Oggier et *al.* in [69], includes self-repairing codes. Self-repairing codes are not maximum distance separable codes and follow the same encoding process as regenerating codes. They minimize the number of nodes to be contacted for repairing one node failure. Encoded fragments can be repaired directly from other subsets of encoded fragments without having to reconstruct first the original data. The number of encoded fragments contacted for repairing is fixed and depends only on how many encoded blocks are missing and not on which specific block is missing.

The previously described codes are block codes. In the context of distributed storage, sparse-graph codes, such as Fountain codes, Raptor codes and LDPC codes can also be used after a convenient arrangement. The next section is dedicated to review Repairable Foutain codes.

2.5 Repairable Fountain codes

Recently, Mackay was the first to suggest Fountain codes for storage and broadcast applications in [64] chapter 50. Thereafter, Asteris and Dimakis proposed distributed Fountain codes in node failure scenarios [3] as a new family of Fountain codes to satisfy efficient repairing in distributed storage. The so-called Repairable Foutain codes are near-optimal erasure correcting codes (near-MDS codes). They are used with a finite number of output nodes in distributed storage context.

In the sequel, we consider repairable Fountain code with k source nodes and n output nodes. The n output nodes represent the n servers where the data is stored. These codes have some desired properties for these applications such as: they are systematic and have logarithmic locality, i.e. an erased output node can be reconstructed by accessing only $O(\log(k))$ other coded symbols. Let us discuss the main properties and the structure of these codes in order to give some insights on the key properties for designing efficient distributed repair codes.

The main properties of repairable Fountain codes are:

• Systematic. The input data is embedded in the encoded output. Consequently, original data can be reconstructed without decoding. From graph theory perspective, k encoded symbols are degree-one. These symbols are called systematic symbols.

• Logarithmic sparsity for parity symbols. Among the n output nodes, k are systematic and the $n-k$ others are parity symbols. Let G denote the generator matrix of repairable Fountain codes. It can be written as $G_{k\times n} = \left[I_{k\times k} \mid P_{k\times n-k}\right]$, where $P_{k\times n-k}$ is the parity matrix and $I_{k\times k}$ the identity matrix which denotes the systematic property. Parity symbols have logarithmic sparsity meaning that $P_{k\times n-k}$ is a sparse matrix,with on average weight $\log(k)$ per column. Therefore, the degree of an encoded symbol is on average $\log(k)$, so each parity symbol is a random linear combination of up to $d(k) = \Omega(\log(k))$ source symbols. It can be demonstrated that $\log(k)$ is the minimal threshold to ensure a successful ML decoding process with a subset of $(1+\epsilon)k$ output symbols.

• Rateless property. This property means that each column is created independently. The number n of columns of G does not have to be specified for the encoder *a priori*.

• Near MDS codes. MDS codes have the property that any subset of k output symbols allows to reconstruct the original data. It means that any subset of k columns of G has

rank k. For repairable Fountain codes, MDS property is relaxed. Indeed, MDS codes have minimum distance $d = n - k + 1$ that is the minimum number of positions in which any two distinct codewords differ. If the considered MDS code is systematic, then the parity matrix P has to be one-matrix, to satisfy the minimum distance.

$$G_{sys_{MDS}} = \left[I_{k\times k} \mid P_{k\times n-k}\right] = \left[\begin{array}{cccc|cccc} 1 & 0 & \cdots & 0 & 1 & 1 & \cdots & 1 \\ 0 & 1 & \cdots & 0 & 1 & 1 & \cdots & 1 \\ \vdots & & \ddots & & \vdots & & \ddots & \\ 0 & \cdots & 0 & 1 & 1 & \cdots & 1 & 1 \end{array}\right] \tag{2.2}$$

Consequently, in order to satisfy sparse parities, repairable Fountain codes cannot be systematic. These codes are near systematic, i.e. a subset of $(1+\epsilon)k$ encoded symbols is needed to reconstruct the original data.

• Edges with coefficients over $\mathbb{F}_q$. Repairable Fountain codes are non-binary codes, i.e. symbols are considered over $\mathbb{F}_q$. A weight f_{ij} is associated to each edge linking an encoded node to an input node, where $f_{ij} \in \mathbb{F}_q$. Note that to achieve small probability of failure as k grows, the size of the field q must grow accordingly.

• ML decoding process. The ML decoding is processed on $(1+\epsilon)k$ encoded symbols. It corresponds to solve a linear system of $(1+\epsilon)k$ equations in $\mathbb{F}_q$. Consequently, decoding probability increases as q increases. The complexity of such decoding is $O(k^3)$.

• Logarithmic locality. The locality of repairable Fountain codes is $d(k) = \log(k)$. Indeed, if a failure occurs among parity symbols, it can be repaired by accessing $d(k) = \log(k)$ systematic symbols. Respectively, if one systematic node is erased, by accessing one parity symbol and $d(k)-1$ systematic symbols it can be repaired. The repair process of one failed node is similar to solving a parity equation. Thus, it is an iterative process. For more than one erasure, the complexity of the repair process is still $\log(k)$ for large value of k.

2.6 Conclusions

In this last section, we have examined the properties of repairable Fountain codes in order to determine the necessary criteria to design efficient repair codes for distributed storage systems. An efficient repair algorithm has to satisfy the following conditions:

- The repair algorithm should not be a complete decoding process. It should be an iterative algorithm.
- The number of neighbors to be contacted should be minimized during the process.

Thus, the key idea is to achieve a compromise between obtaining good locality and iterative decoding process. In fact, good locality can be obtained through the design of

sparse parity-check matrix. On the other hand, the code design should allow for an iterative decoding process in order to reduce the high complexity of ML decoding.
For these reasons, others codes presented in Chapter 1 as polar codes and Reed-Muller codes are not suitable for distributed storage applications. In fact, the repair process of a polar code is a complete decoding and their locality is too high though they are capacity achieving codes. The locality of Reed-Muller codes is adapted to distributed storage applications. Consider an $R(r, m)$ code, its locality is $2^{r+1} - 1$ [47]. RM codes achieve capacity on the BEC under MAP decoding [57][58]. Unfortunately, a MAP decoding is too complex for distributive storage applications.

This issue has motivated us to study spatially coupled LDPC codes, that can achieve maximum-a-posteriori decoding threshold under iterative Belief Propagation (BP) decoding. This study is the subject of Chapter 3. On the other hand, existing codes in the literature have been developed to deal only with a few number of machines (nodes) erased or unavailable. No coding scheme is proposed to handle a cluster failure. Chapters 3 & 4 use the diversity criterion in order to design new coding schemes that are resilient to cluster erasure in addition to i.i.d. symbol erasures.

Chapter 3

Spatial Coupling for Codes and Diversity

Linear block codes and convolutional codes were developed by two separate communities as witnessed in the history of channel coding. Viterbi and Omura reported a higher error exponent for time-varying convolutional codes, see section 5.2 in [112]. Another major success of convolutional codes is found in the construction of parallel turbo codes [8] and its counterpart is found in [36] for LDPC codes. In recent coding theory, both linear block codes and convolutional codes are studied or represented in similar ways leading to new structures such as LDPC convolutional codes [34], Tanner convolutional codes [111], and quasi-cyclic block and convolutional codes [102]. Exceptional thresholds of LDPC convolutional codes [23][60] led to the discovery of threshold saturation [55] and to the general concept of spatial coupling [56]. Spatial coupling is currently applied in many areas in coding and information theory. The wide application of spatial coupling ranges from coding for the wiretap channel [81] to coding for the multiple access channel [118]. Generally coupled structures including multiple chains and loops have been proposed by Truhachev et al. [106][107]. Their connected spatially-coupled chains aim at improving the iterative decoding threshold and minimizing the decoding complexity.

A spatially-coupled LDPC ensemble is formed by multiple LDPC ensembles sharing common edges. The whole coupled structure resembles a convolutional code with extra check nodes on both extremal sides [55] playing the role of a trellis termination. The parity-check description of a coupled ensemble is similar to LDPC convolutional codes originally proposed by Felström and Zigangirov [34]. Trellis termination for a chain of coupled ensembles has a direct impact on the threshold saturation and on the finite-length performance. Chain loops is another way of improving the threshold instead of terminating the chain on both sides [106]. For asymptotic coding schemes, the advantages of spatially-coupled ensembles can be recapitulated in three main points: regular degree distributions substituting for complex irregular distributions, iterative belief propagation decoding achieving maximum-a-posteriori performance, and a universal spatial coupling concept that can be applied in many areas beyond LDPC coding [36].

The theory of error-correcting codes was limited to channels with errors and erasures that are independently located inside a codeword [65], i.e. ergodic channels. Similarly, all channels considered in modern coding theory are also ergodic in nature [36][7][86]. The non-ergodic fading channel encountered in wireless communications requires a special coding approach [108][9]. Root-LDPC codes were designed to deal with non-ergodic (quasi-static) fading and attain a maximal diversity order allowed by the number of degrees of freedom in the block-fading channel and authorized by the Root-LDPC coding rate. The interest in applying spatial coupling to Root-LDPC codes is motivated by the use of anti-Root LDPC ensembles for secure communications on multiple-link channels [26][18].

The chapter is structured as follows. Section 3.1 gives a brief overview of uniform spatial coupling as known in the literature. The forward-layered LDPC coupling method is presented in Section 3.2. Non-uniform chains of spatial coupling and spatially-variant spatial coupling are also introduced in section 3.2. To improve the threshold boundary of Root-LDPC, we propose a new spatial coupling structure based on parity bits doping in Section 3.3. Finally, in Section 3.4, spatially-coupled Root-LDPC is applied to distributed storage and diversity applications.

3.1 State of the art: uniform spatial coupling

This section gives a very brief overview of uniform spatial coupling, i.e. the standard coupling as known in the literature. More details have been presented in Chapter 1.7. Consider a (d_b, d_c)-regular binary LDPC ensemble with M_b variable nodes of degree d_b and $\frac{d_b}{d_c}M_b$ check nodes of degree d_c. Edges connecting variable nodes and check nodes are given by a random matching between the $d_b \times M_b$ sockets of variable nodes and the $d_b \times M_b$ sockets of check nodes, see section 3.4 in [86]. It is assumed that M_b is infinitely large. This ensemble is referred to as the *uncoupled ensemble*. The design rate is $1 - \frac{d_b}{d_c}$ and $\frac{d_b}{d_c} = h^S$ is the Shannon threshold [56].
Building a coupled (d_b, d_c, L_c, w) ensemble works as follows. Define L_c spatial positions on a horizontal line. On each spatial position i, $i = 1 \dots L_c$, place M_b variable nodes and $\frac{d_b}{d_c}M_b$ check nodes. Extra check nodes can be added at positions $i < 1$ and $i > L_c$ for the purpose of left and right termination of the coupled chain. The d_b edges of a variable node at position i are assumed to be uniformly and independently connected to check nodes in the range $[i, i + w - 1]$. The d_c edges of a check node at position i are assumed to be uniformly and independently connected to variable nodes in the range $[i - w + 1, i]$. The smoothing parameter w can be seen as the memory of LDPC convolutional codes built from the (d_b, d_c, L_c, w) coupled ensemble. Such coupling is called *uniform* for two reasons, connections are uniform in $[i, i + w - 1]$ and the L_c uncoupled ensembles are all regular with identical degrees.

On BMS channels, two key results have been recently proven. The first one stated

in Theorem 41 in [56] is

$$\lim_{w\to\infty}\lim_{L_c\to\infty} h^{BP}_{coupled} \;=\; h^{A}_{uncoupled}, \tag{3.1}$$

where h^{BP} is the threshold under belief propagation BP decoding and h^A is the area threshold ($h^A = h^{MAP}$ for the BEC). The second result stated in Lemma 29 in [56] is, for a given design rate,

$$\lim_{d_b\to\infty} h^{A}_{uncoupled} \;=\; h^{S}. \tag{3.2}$$

In the next section, we restrict our study to spatial coupling with a short memory $w = 2$ and the BEC is the default transmission channel. All results are extendable to binary memoryless symmetric (BMS) channels.

3.2 Forward-layered LDPC coupling

In this section, we propose a new method for spatial coupling. Firstly, our method considers a chain of uncoupled LDPC ensembles, then new edges are added to the chain to couple an ensemble with its neighbor on one side. This method, referred to as forward-layered LDPC coupling, is inspired from unequal error protection UEP [19][45]. Indeed, our initial objective is to create a limited number of classes for UEP [19][45], e.g. three classes of digits with decreasing error rate performance to imitate graceful degradation encountered in analog systems. UEP can be carried out via layered coding. Layered coding has been recently proposed for the Gaussian interference channel [103]. Introducing a large number of levels in layered coding yields our method of forward-layered LDPC coupling. Another novelty is the definition of a degree distribution for the spatial coupling. Edges connecting two neighboring ensembles are not part of their protographs, these spatial coupling edges are defined by an extra degree distribution.

3.2.1 Construction of the forward-layered LDPC coupling

Layered coding is usually made out via binning or superposition [103]. Another simple method to build layered codes for UEP is overlapping. The output of a first encoder is partially or fully fed to the input of a second encoder. The process is repeated for a bunch of L_c encoders. An example of overlapped layered coding is shown in Figure 3.1.

Firstly, a direct sum of L_c regular $(3, 6)$ LDPC codes is constructed by copying the $(3, 6)$ parity-check matrix on the main diagonal. LDPC codes properties are detailed in Chapter 1.5. Then, each $(3, 6)$ parity-check matrix is connected to its next neighbor via a $(1, 2)$ sparse matrix, i.e. a random sparse binary matrix with weight one per column and weight two per row. The parity-check structure in Figure 3.1 is very similar to a coupled $(4, 8, L_c, w = 2)$ ensemble but it differs in how edges connect neighboring copies. More precisely, for a window length $w = 2$, forward-layered coupling allows us to introduce a new non-uniformly coupled ensemble denoted by its parameters

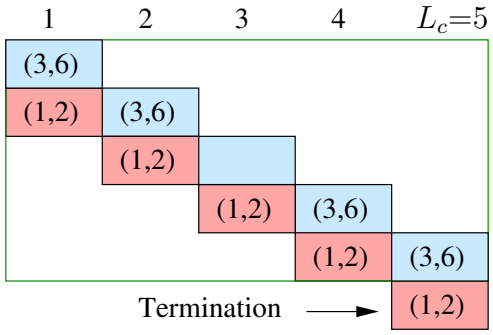


Figure 3.1: Parity-check matrix of overlapped L_c-layered coding built from $(3, 6)$-regular LDPC ensembles coupled via $(1, 2)$ sparse binary matrices.

$(d_b, d_c; d_{bs}, d_{cs}; L_c)$. This new ensemble is an alternative to the uniform coupling ensemble $(d_b + d_{bs}, d_c + d_{cs}, L_c, w = 2)$.

Construction of the coupled $(d_b, d_c; d_{bs}, d_{cs}; L_c)$ **ensemble:** Define L_c spatial positions on a horizontal line. On each spatial position i, $i = 1 \dots L_c$, place M_b variable nodes and $\frac{d_b}{d_c} M_b$ check nodes. Extra check nodes can be added at position $i > L_c$ for the purpose of right termination of the coupled chain. The d_b edges of a variable node at position i are assumed to be connected to check nodes in position i. The d_c edges of a check node at position i are assumed to be connected to variable nodes in position i. Spatial forward-layered coupling is composed by extra spatial coupling edges. Each variable node at position i has extra d_{bs} edges connected to check nodes in position $i + 1$. Each check node at position i has extra d_{cs} edges connected to variable nodes in position $i - 1$. The (d_{bs}, d_{cs}) spatial coupling should satisfy the layered coding constraint $\frac{d_{bs}}{d_{cs}} = \frac{d_b}{d_c}$ translating the fact that both (d_b, d_c) and (d_{bs}, d_{cs}) matrices are of same size. The structure in Figure 3.1 corresponds to a $(3, 6; 1, 2; L_c)$ ensemble. Its compact graph representation (with protographs) is depicted in Figure 3.2.

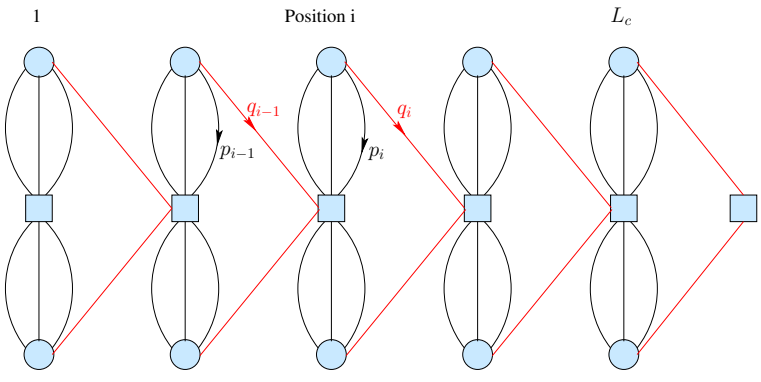


Figure 3.2: Tanner graph for $(3, 6)$-regular forward-layered spatial coupling.

Without chain termination, the design rate for the $(d_b, d_c; d_{bs}, d_{cs}; L_c)$ ensemble is $1 - \frac{d_b}{d_c}$. The forward coupling structure has no left termination. A chain termination can be placed on the right of the coupled chain, i.e. an extra (d_{bs}, d_{cs}) local ensemble with its check nodes at $i = L_c + 1$. The rate loss due to termination is a factor of $1 - 1/L_c$. Other types of chain termination are possible for uniform and forward coupling, they will not be discussed in this chapter.

Density evolution (DE) equations for a uniform spatially coupled (d_b, d_c, L_c, w) ensemble over the BEC(ϵ) are [55]

$$p_i = \epsilon \left(1 - \frac{1}{w} \sum_{j=0}^{w-1} \left(1 - \frac{1}{w} \sum_{k=0}^{w-1} p_{i+j-k} \right)^{d_c - 1} \right)^{d_b - 1} \tag{3.3}$$

for $i = 1 \dots L_c$. For $w = 2$, right termination is incorporated by defining $p_{L_c+1} = 0$. DE fixed-point equations for a spatially coupled $(d_b, d_c; d_{bs}, d_{cs}; L_c)$ ensemble require more messages because of the multiple edge types. Let p_i denote the message from variable node to check node in position i and q_i the ongoing message to check node in position $i + 1$.

Proposition 3.1. *Let d_b, d_c, d_{bs}, and d_{cs} be positive integers satisfying $\frac{d_{bs}}{d_{cs}} = \frac{d_b}{d_c}$. DE equations for a forward-layered spatially coupled $(d_b, d_c; d_{bs}, d_{cs}; L)$ ensemble are*

$$\begin{aligned} p_i &= \epsilon \cdot f_{i+1}^{d_{bs}} \cdot \left(1 - (1 - p_i)^{d_c - 1} (1 - q_{i-1})^{d_{cs}} \right)^{d_b - 1}, \\ q_i &= \epsilon \cdot f_{i+1}^{d_{bs} - 1} \cdot \left(1 - (1 - p_i)^{d_c - 1} (1 - q_{i-1})^{d_{cs}} \right)^{d_b}, \\ f_{i+1} &= \left(1 - (1 - p_{i+1})^{d_c} (1 - q_i)^{d_{cs} - 1} \right), \end{aligned}$$

for $i = 1 \dots L_c$, where f_{i+1} is a check-to-bit message propagating backward from position $i + 1$ to position i.

Proof: From the compact graph representing the coupled chain in Figure 3.2, a reader who is familiar with modern coding theory [86] can quickly check that DE equations for the $(3, 6; 1, 2; L_c)$ ensemble are

$$\begin{aligned} p_i &= \epsilon \cdot f_{i+1} \cdot \left(1 - (1 - p_i)^5 (1 - q_{i-1})^2 \right)^2, \\ q_i &= \epsilon \cdot \left(1 - (1 - p_i)^5 (1 - q_{i-1})^2 \right)^3, \\ f_{i+1} &= \left(1 - (1 - p_{i+1})^6 (1 - q_i) \right). \end{aligned}$$

Then, it is straightforward to generalize to any integer parameters defining the coupled ensemble. The equal ratios between the degrees of the local ensemble and the degrees

of the spatial coupling forces variable nodes and check nodes to have equal number of sockets for connecting edges. *QED.*

Using (3.3) and Proposition 3.1, belief propagation thresholds are computed on the BEC for different rate-1/2 ensembles. Thresholds are listed in Table 3.1 for $d_{bs} = 1$ and $d_{cs} = 2$. A uniform spatially coupled $(d_b, d_c, L_c, w = 2)$ ensemble (column 4) is compared to a forward-layered $(d_b - 1, d_c - 2; 1, 2; L_c)$ ensemble (column 5). For both ensembles, right chain termination is made by $p_{L_c+1} = 0$. Any chain length L_c sufficiently large yields same threshold results, typical values are $L_c = 60$ or $L_c = 100$. Table 3.1 shows that forward-layered LDPC coupling outperforms the standard uniform spatial coupling.

Ensemble	$h^{BP}_{uncoupled}$	$h^{MAP}_{uncoupled}$	$h^{BP}_{uniform}$	$h^{BP}_{forward}$
(3,6)	0.42944	0.48815	0.48808	0.48815
(4,8)	0.38345	0.49774	0.49442	0.49741
(5,10)	0.34155	0.49948	0.48268	0.49811
(6,12)	0.30746	0.49988	0.46031	0.49667

Table 3.1: BEC thresholds of uniform spatial coupling versus forward-layered coupling, $w = 2$, $d_{bs} = 1$, and $d_{cs} = 2$.

Another illustration of the forward spatial coupling performance is given in Figure 3.3. BEC thresholds are plotted versus the spatial position i/L_c, for $i = 1 \dots L_c$, for the $(3, 6; 1, 2; L_c)$ ensemble. The coupled ensemble with termination largely overpasses the thresholds of the uncoupled $(3, 6)$ and $(4, 8)$ LDPC. In fact, it saturates at 0.49741 very close to the $(4, 8)$-LDPC MAP threshold, see the $(4, 8)$ row in Table 3.1. The small value $L_c = 10$ is used for the purpose of illustration, its rate loss makes it unacceptable in practice. Without chain termination, forward-layered coupling improves the threshold but stays blocked at a relatively small value of 0.44695. Uniform coupling is capable of saturating the threshold without chain termination but unfortunately, in the case of $w = 2$, it may need up to 6 spatial positions to push up the threshold from right to left starting at $i = L_c$. This loss in local thresholds cannot be tolerated for finite L_c. Hence, we compare both coupled ensembles under right termination for a typical finite value of chain length L_c.

Results in Figure 3.3 are found for 10000 decoding iterations. In order to make the threshold flat with respect to the spatial position i/L_c, the number of decoding iterations increases with L_c. Furthermore, the $w = 2$ forward spatial coupling lets a (d_b, d_c) ensemble at position i strengthen its neighbor at position $i - 1$, i.e. variable nodes protection propagates from right to left due to the structure of the (d_{bs}, d_{cs}) spatial coupling. Indeed, coupling edges leave a check node at position i to reach a variable node at position $i - 1$.

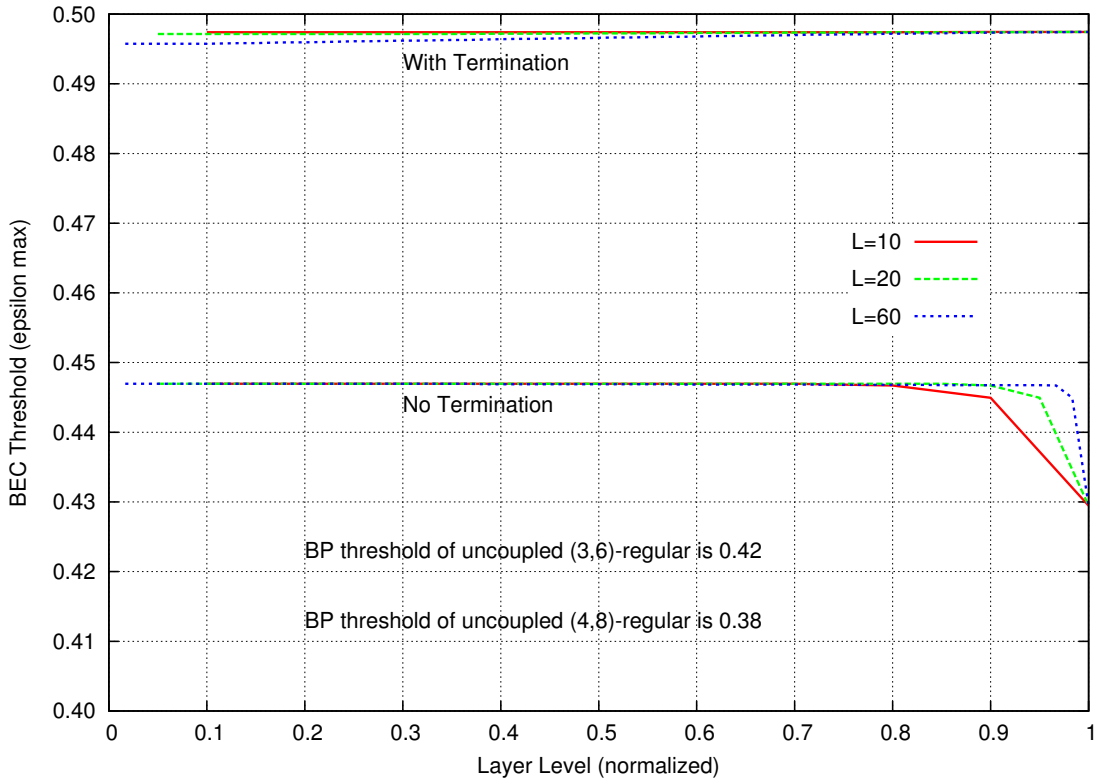


Figure 3.3: Performance of the forward coupled $(3, 6; 1, 2; L_c)$ ensemble.

3.2.2 Non-uniform chain of spatial coupling

In this section, LDPC ensembles with different parameters d_b and d_c are coupled. The local (d_b, d_c) ensemble is kept regular (i.e. d_b and d_c are integer), but the (d_{bs}, d_{cs}) spatial coupling is not necessarily regular. The objective is to create a stair of thresholds for unequal error protection. Following the same arguments and notations as previously, let us build a non-uniform chain by gluing non-identical LDPC ensembles.

Definition 3.1. Consider two coupled ensembles $(d_b, d_c; d_{bs}, d_{cs}; L_{c_1})$ and $(d'_b, d'_c; d'_{bs}, d'_{cs}; L_{c_2})$ corresponding to two chains of length L_{c_1} and L_{c_2} respectively. The new ensemble defined by gluing the two chains is written as

$$(d_b, d_c; d_{bs}, d_{cs}; L_{c_1}) \ \oplus \ (d'_b, d'_c; d'_{bs}, d'_{cs}; L_{c_2})$$

where the new chain has length $L_c = L_{c_1} + L_{c_2}$.

In general, a chain of length $L_c = \aleph L_{c_1}$ is built by gluing $\aleph$ sub-chains. The new coupled ensemble can be written as

$$\bigoplus_{j=1}^{\aleph} (d_b(j), d_c(j); d_{bs}(j), d_{cs}(j); L_{c_1}). \tag{3.4}$$

In this case, $d_b(j)$ and $d_c(j)$ are integers but not necessarily $d_{bs}(j)$ and $d_{cs}(j)$, i.e. the spatial coupling is irregular. Long and complex degree distributions are unnecessary. We will define and use the simplest irregular distributions.

Definition 3.2. $\Psi(\alpha, d)$ is a parametric polynomial that includes two monomials as follows

$$\Psi(\alpha, d) = \alpha x^d + (1-\alpha)x^{d+1}.$$

Let $d_{bs}(j)$ and $d_{cs}(j)$ be two positive real numbers. The real $d_{bs}(j)$ is the average degree of variable nodes at the left of a spatial coupling connecting two protographs of type $(d_b(j), d_c(j))$ inside sub-chain j. Similarly, $d_{cs}(j)$ is the average degree of check nodes at the right of a spatial coupling. Instead of showing a sub-chain section, Figure 3.4 displays a general section in the total chain between spatial positions i and $i+1$. If position i belongs to sub-chain j then, for example, $d_{b,i} = d_b(j)$ and $d_{bs,i} = d_{bs}(j)$. From a node perspective, the left and right degree distributions of the spatial coupling are taken to be

$$\begin{aligned} \overset{\circ}{\lambda}_s(x) &= \Psi(\lfloor d_{bs} \rfloor + 1 - d_{bs}, \lfloor d_{bs} \rfloor), \\ \overset{\circ}{\rho}_s(x) &= \Psi(\lfloor d_{cs} \rfloor + 1 - d_{cs}, \lfloor d_{cs} \rfloor). \end{aligned} \tag{3.5}$$

In multiple-edge-type LDPC structures, such as [16][17], both node and edge perspectives are required in DE equations. From an edge perspective, left and right degree distributions associated to (3.5) are

$$\begin{aligned} \lambda_s(x) &= \Psi(\tfrac{\lfloor d_{bs} \rfloor}{d_{bs}}(\lfloor d_{bs} \rfloor + 1 - d_{bs}), \lfloor d_{bs} \rfloor - 1), \\ \rho_s(x) &= \Psi(\tfrac{\lfloor d_{cs} \rfloor}{d_{cs}}(\lfloor d_{cs} \rfloor + 1 - d_{cs}), \lfloor d_{cs} \rfloor - 1). \end{aligned} \tag{3.6}$$

From (3.5) or (3.6), the reader can easily check that given distributions have average degrees d_{bs} and d_{cs}. The non-uniform chain of length $L_c = \aleph L_{c_1}$ comprises $\aleph$ pairs $(d_{bs}(j), d_{cs}(j))$, for $j = 1 \dots \aleph$.

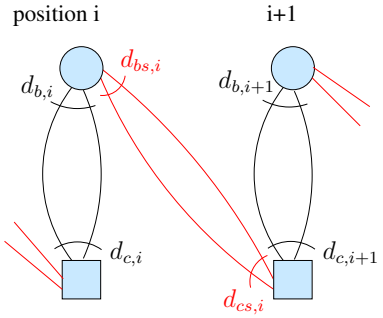


Figure 3.4: Protographs with $(d_{bs,i}, d_{cs,i})$ spatial coupling.

Density evolution fixed-point equations of Proposition 3.1 are now updated by incorporating polynomials from (3.5) and (3.6). As a well-known rule, alike polynomials

used in DE equations for Root-LDPC ensembles [16][17], edge perspective is applied on messages located on edges at the same side from a node generating the DE message. Node perspective is applied on messages located on edges at the other side from that node. In the chain defined by (3.4), DE equations at spatial position i are given by the following proposition.

Proposition 3.2. *For $i = 1 \dots L_c$, we have*

$$p_i = \epsilon \cdot \mathring{\lambda}_s(f_{i+1}) \cdot \left(1 - (1-p_i)^{d_c-1} \cdot \mathring{\rho}_s(1-q_{i-1})\right)^{d_b-1},$$

$$q_i = \epsilon \cdot \lambda_s(f_{i+1}) \cdot \left(1 - (1-p_i)^{d_c-1} \cdot \mathring{\rho}_s(1-q_{i-1})\right)^{d_b},$$

$$f_{i+1} = 1 - (1-p_{i+1})^{d_c} \cdot \rho_s(1-q_i),$$

where $d_b = d_{b,i} \in \mathbb{N}$, $d_c = d_{c,i} \in \mathbb{N}$, $d_{bs} = d_{bs,i} \in \mathbb{R}^+$, and $d_{cs} = d_{cs,i} \in \mathbb{R}^+$ with the following layered coding constraint $\frac{d_{bs,i}}{d_{cs,i}} = \frac{d_{b,i+1}}{d_{c,i+1}}$, i.e. the degree ratio for a spatial coupling at position i is imposed by the protograph at position $i+1$.

A stair for unequal error protection is plotted on Figure 3.5. The five uncoupled LDPC ensembles are, from left to right, $(3, 12)$, $(3, 9)$, $(3, 6)$, $(3, 5)$, and $(3, 4)$. The corresponding spatial coupling is $(1, 4)$, $(1, 3)$, $(1, 2)$, $(1.2, 2)$, and $(1.5, 2)$. The threshold stair is saturating up on Shannon stair defined by the five local Shannon thresholds.

3.2.3 Spatially-variant spatial coupling

Let us consider the highest number of sub-chains inside a non-uniform chain. This extremal situation corresponds to $\aleph = L_c$. The local LDPC ensemble has real average degrees $d_{b,i}$ and $d_{c,i}$ varying with respect to i, for $i = 1 \dots L_c$. The irregularity of left and right degree distributions is taken into account by $\mathring{\lambda}(x)$, $\lambda(x)$, $\mathring{\rho}(x)$, and $\rho(x)$. These polynomials are defined in a similar fashion as in (3.5) and (3.6). For example, we have $\mathring{\lambda}(x) = \Psi(\lfloor d_b \rfloor + 1 - d_b, \lfloor d_b \rfloor)$. The spatially-variant chain becomes

$$\bigoplus_{i=1}^{L_c} (d_{b,i}, d_{c,i}; d_{bs,i}, d_{cs,i}; 1). \tag{3.7}$$

Proposition 3.3. *Let $d_{b,i} \in \mathbb{R}^+$, $d_{c,i} \in \mathbb{R}$, $d_{bs,i} \in \mathbb{R}$, $d_{cs,i} \in \mathbb{R}$, satisfy $\frac{d_{bs,i}}{d_{cs,i}} = \frac{d_{b,i+1}}{d_{c,i+1}}$, for $i = 1 \dots L_c$. DE equations for this spatially-variant ensemble are*

$$p_i = \epsilon \cdot \mathring{\lambda}_{s,i}(f_{i+1}) \cdot \lambda_i \left(1 - \rho_i(1-p_i) \cdot \mathring{\rho}_{s,i-1}(1-q_{i-1})\right),$$

$$q_i = \epsilon \cdot \lambda_{s,i}(f_{i+1}) \cdot \mathring{\lambda}_i \left(1 - \rho_i(1-p_i) \cdot \mathring{\rho}_{s,i-1}(1-q_{i-1})\right),$$

$$f_{i+1} = 1 - \mathring{\rho}_{i+1}(1-p_{i+1})\rho_{s,i}(1-q_i).$$

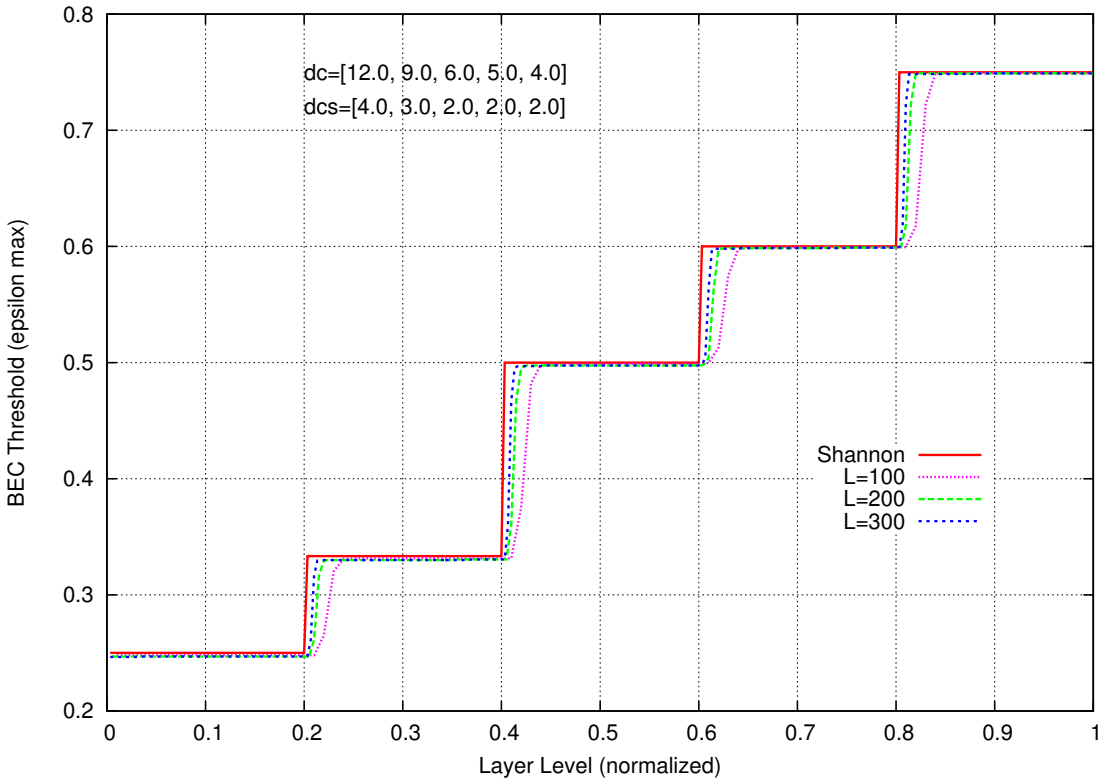


Figure 3.5: BEC thresholds for a non-uniform chain of forward spatial coupling with $\aleph = 5$ sub-chains each of length $L_{c_1} = L_c/5$.

Threshold results for a spatially-variant chain are plotted in Figure 3.6 for the binary erasure channel and $L_c = 200$.

3.3 Spatial coupling of Root-LDPC ensembles: Parity bits doping

In this section, we propose to apply forward-layered coupling to Root-LDPC codes and study the effect of spatial coupling on their outage boundary in the fading plane. Root-LDPC codes are low-density parity-check codes specifically designed to tackle with non-ergodic (quasi-static) fading [17][16]. The key idea is to bring more structure to a random LDPC ensemble in order to let information variable nodes receive BP messages including all available fading states. Hence, a Root-LDPC ensemble attains a diversity order which is equal to the number of degrees of freedom in the block-fading channel.

3.3.1 Main properties of Root-LDPC ensembles

Let us recall the Root-LDPC construction and introduce the notion of diversity.

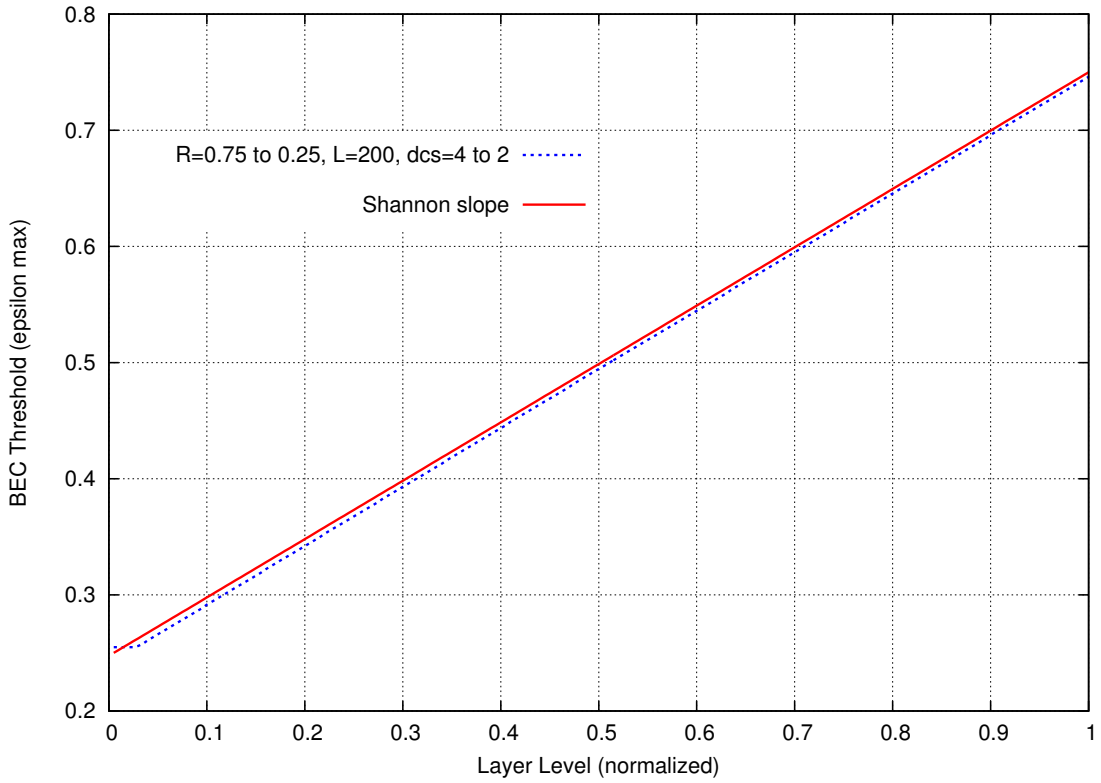


Figure 3.6: BEC thresholds for a non-uniform chain of forward spatial coupling with $\aleph = L_c$ local ensembles varying with the spatial position. Parameters are $L_c = 200$, $d_b = 3$, $d_c = 12$ to 4, $Rate = 0.75$ to 0.25.

Root-LDPC ensembles are multi-edge-type LDPC ensembles with specific properties [17]. For simplicity, let us consider an introductory example with rate-1/2 LDPC code of length N, where N bits are transmitted on a block-fading channel with $M = 2$ fading states (denoted by colors in the next Section). For this rate-1/2 ensemble four classes of bits are defined by construction:

• On the first channel (ϵ_1): $N/4$ information bits referred to as $1i$ and $N/4$ parity bits referred to as $1p$.

• On the second channel fading (ϵ_2): $N/4$ information bits referred to as $2i$ and $N/4$ parity bits referred to as $2p$.

In the same way, for the rate-1/2 ensemble, there are two classes of check nodes named $1c$ and $2c$. Check nodes of type $1c$ are rootchecks for $1i$ and check nodes $2c$ are rootchecks for $2i$. The notion of a *rootcheck* is defined as follows.

Definition 3.3. Consider a check node with a root variable node transmitted on a fading channel ϵ_i. This check node is a *rootcheck* with respect to its root if its leaf variable nodes undergo fadings ϵ_j, where $j \neq i$.

This rate-1/2 Root-LDPC code is full-diversity under iterative message passing when transmitted on a 2-state block-erasure channel. Diversity is encountered in different communication systems according to two principal aspects:
• Diversity from error-control coding point of view. It consists in creating many replicas of the same information.
• Diversity from communication channel point of view. It corresponds to the number of available degrees of freedom while transmitting information.
This code is also MDS according to the block-fading Singleton bound and has the highest coding rate that attains a double diversity (see Sec. V in [16]). Under ML decoding, an upper bound on the diversity order L is given by the block-fading Singleton Bound [51] [66],

$$L \leq 1 + \lfloor M(1-R) \rfloor, \tag{3.8}$$

where R is the coding rate and M is the intrinsic channel diversity. For rate $R = 1/M$, Root-LDPC codes can attain a full-diversity order $L = M$ [16]. $R = 1/M$ is the maximal achievable coding rate for $L = M$ according to (3.8). The coding rate can exceed $1/M$ when $L < M$. As an example for distributed storage applications, rate-3/4 Root-LDPC codes yield $L = 2$ when $M = 4$.

In order to guarantee full diversity for information bits (double diversity is the maximal diversity for rate 1/2 with two channel states), information and parity bits are connected to checks $1c$ and $2c$ as shown by the compact Tanner graph of the rate-1/2 Root-LDPC ensemble in Figure 3.7. Edges between $1i$ and $1c$ are given by a graph matching which is a permutation matrix in matrix representation.

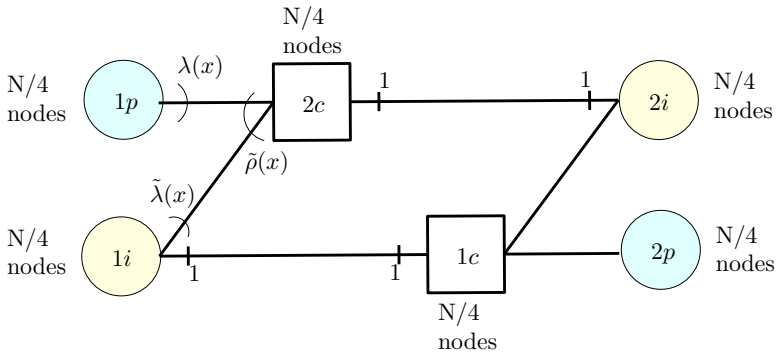


Figure 3.7: Compact Tanner graph representation for a rate-1/2 Root-LDPC ensemble. If the bits transmitted on one fading are erased, the erased information bits are recovered.

Density evolution of uncoupled Root-LDPC ensembles

Density evolution equations of the information and parity messages on two parallel BECs for the uncoupled Root-LDPC ensembles are summarized here [17]. Consider the channel model in Figure 3.8, where N bits are transmitted on two parallel BECs with the

erasure probabilities ϵ_1 and ϵ_2. $N/2$ bits are transmitted on BEC(ϵ_1) and $N/2$ bits are transmitted on BEC(ϵ_2).

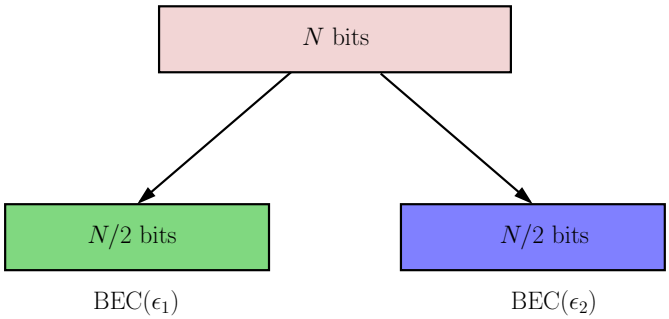


Figure 3.8: Parallel binary erasure channels where half of the bits are transmitted through BEC(ϵ_1) and the the other half of the bits are transmitted through BEC(ϵ_2).

Define the degree distribution polynomials as

$$\lambda(x) = \sum_{i=2}^{d_b} \lambda_i x^{i-1} \qquad \rho(x) = \sum_{j=2}^{d_c} \rho_j x^{j-1} \tag{3.9}$$

The definitions of the degree polynomials $\mathring{\lambda}(x)$, $\mathring{\rho}(x)$, $\tilde{\lambda}(x)$, and $\tilde{\rho}(x)$ are as in [17]. Let us define the six message densities for a Root-LDPC ensemble as follows:

- f_1 is the density of message $1i \to 2c$.
- q_1 is the density of message $1i \to 1c$.
- f_2 is the density of message $2i \to 1c$.
- q_2 is the density of message $2i \to 2c$.
- g_1 is the density of message $1p \to 2c$.
- g_2 is the density of message $2p \to 1c$.

DE equations for the six message densities q_1^m, f_1^m, g_1^m, q_2^m, f_2^m, and g_2^m can be found by drawing the local neighborhood of each type of variable nodes at decoding iteration $m+1$ as:

$$q_1^{m+1} = \epsilon_1 \mathring{\lambda}\left(1 - (1 - q_2^m)\,\tilde{\rho}\left(1 - f_e f_1^m - g_e g_1^m\right)\right), \tag{3.10}$$

$$f_1^{m+1} = \epsilon_1 \tilde{\lambda}\left(1 - (1 - q_2^m)\,\tilde{\rho}\left(1 - f_e f_1^m - g_e g_1^m\right)\right)\left(1 - \mathring{\rho}\left(1 - f_e f_2^m - g_e g_2^m\right)\right), \tag{3.11}$$

$$g_1^{m+1} = \epsilon_1 \lambda\left(1 - (1 - q_2^m)\,\tilde{\rho}\left(1 - f_e f_1^m - g_e g_1^m\right)\right), \tag{3.12}$$

$$q_2^{m+1} = \epsilon_2 \mathring{\lambda}\left(1 - (1 - q_1^m)\,\tilde{\rho}\left(1 - f_e f_2^m - g_e g_2^m\right)\right), \tag{3.13}$$

$$f_2^{m+1} = \epsilon_2 \tilde{\lambda}\left(1 - (1 - q_1^m)\,\tilde{\rho}\left(1 - f_e f_2^m - g_e g_2^m\right)\right)\left(1 - \mathring{\rho}\left(1 - f_e f_1^m - g_e g_1^m\right)\right), \tag{3.14}$$

$$g_2^{m+1} = \epsilon_2 \lambda\left(1 - (1 - q_1^m)\,\tilde{\rho}\left(1 - f_e f_2^m - g_e g_2^m\right)\right), \tag{3.15}$$

where the multi-edge-type fractions are defined as functions of the average variable-node degree $\bar{d}_b$:

$$f_e = 1 - g_e = \frac{\bar{d}_b - 1}{2\bar{d}_b - 1}. \tag{3.16}$$

For regular ensembles with variable-node degree d_b and check-node degree d_c, $\mathring{\lambda}(x) = \lambda(x)$, $\mathring{\rho}(x) = \rho(x)$, $\tilde{\lambda}(x) = \lambda(x)/x$, and $\tilde{\rho}(x) = \rho(x)/x$. Consequently, $q_1(x) = g_1(x)$ and $q_2(x) = g_2(x)$. Then, we have four density evolution equations, and $\lambda(x)$ and $\rho(x)$ are monomials:

$$\lambda(x) = x^{d_b - 1} \qquad \rho(x) = x^{d_c - 1} \tag{3.17}$$

Then, DE equations for BEC(ϵ_1) and BEC(ϵ_2) can be written as:

$$q_1^{m+1} = \epsilon_1 \left(1 - (1 - q_2^m)\,(1 - f_e f_1^m - g_e q_1^m)^{d_c - 2}\right)^{d_b - 1}, \tag{3.18}$$

$$f_1^{m+1} = \epsilon_1 \left(1 - (1 - q_2^m)\,(1 - f_e f_1^m - g_e q_1^m)^{d_c - 2}\right)^{d_b - 2} \left(1 - (1 - f_e f_2^m - g_e q_2^m)^{d_c - 1}\right), \tag{3.19}$$

$$q_2^{m+1} = \epsilon_2 \left(1 - (1 - q_1^m)\,(1 - f_e f_2^m - g_e q_2^m)^{d_c - 2}\right)^{d_b - 1}, \tag{3.20}$$

$$f_2^{m+1} = \epsilon_2 \left(1 - (1 - q_1^m)\,(1 - f_e f_2^m - g_e q_2^m)^{d_c - 2}\right)^{d_b - 2} \left(1 - (1 - f_e f_1^m - g_e q_1^m)^{d_c - 1}\right). \tag{3.21}$$

Recall that for Root-LDPC, only the information bits are connected to rootchecks and have full-diversity. Thus, the threshold bound is determined by the convergence of the densities f_1 and f_2 associated with the messages $1i \to 2c$ and $2i \to 1c$ respectively.

As we show later in Figure 3.15, the gap between the threshold boundary of Root-LDPC and the capacity boundary of the channel is not small enough.

3.3.2 Spatial coupling of Root-LDPC ensembles

In order to improve the coding gain, we propose a spatial coupling structure for Root-LDPC based on the coupling structure in Section 3.2 for random LDPC ensembles.

Construction of the partially-coupled Root-LDPC ensemble: L_c copies of (d_b, d_c) Root-LDPC ensembles are placed in spatial positions $l = 1, 2, ..., L_c$ to form a chain of length L_c. Then, extra edges are added to couple the ensembles. In order to simplify the construction, only parity variable nodes are coupled to the check nodes in the next spatial position resulting in a partial-coupling scheme as shown in Figure 3.9. To our knowledge, this is the first time partial coupling is applied to LDPC ensembles for parity bits doping. For termination at the right end, extra check nodes are placed at the $(L_c + 1)$-Th. spatial position.

In addition to the local message densities defined in Section 3.3.1, we have four more coupling messages. Let us define the coupling message densities as follows:

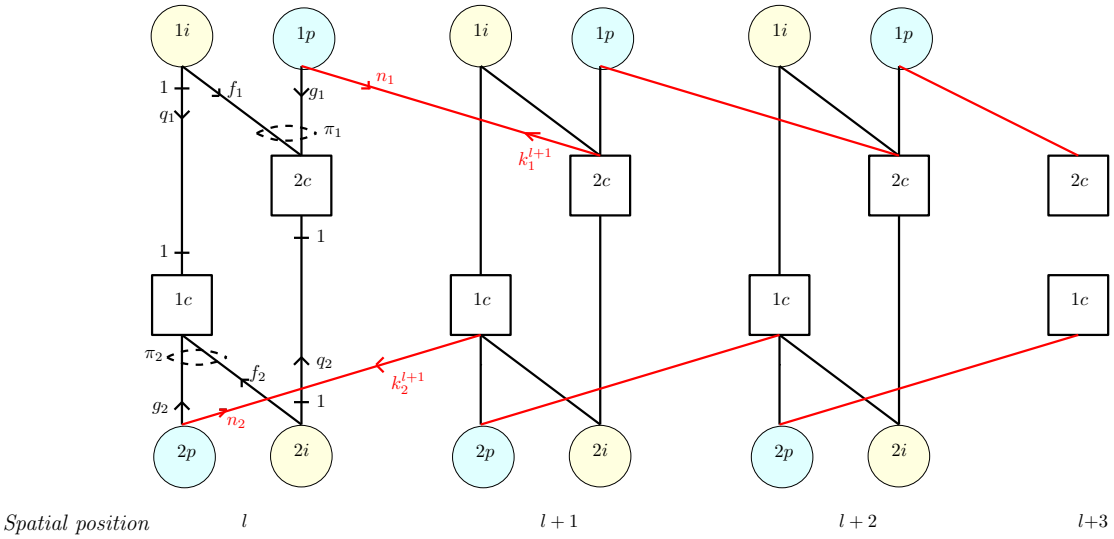


Figure 3.9: Forward-layered spatial coupling of Root-LDPC. The chain is terminated at the right end by the checks placed at the spatial position $L_c + 1$. Since only the parity bits are coupled to the checks, this is a partial coupling scheme.

- n_1 is the density of message $1p(l) \to 2c(l+1)$.
- n_2 is the density of message $2p(l) \to 1c(l+1)$.
- k_1^{l+1} is the density of message propagating backward $2c(l+1) \to 1p(l)$.
- k_2^{l+1} is the density of message propagating backward $1c(l+1) \to 2p(l)$.

Proposition 3.4. *Let (d_b, d_c) denote the degrees of variable nodes and check nodes of the uncoupled regular LDPC ensemble and (d_{bs}, d_{cs}) denote the degrees of the extra edges for coupling the parity variable nodes and the checks. Then, DE equations for forward-layered spatially coupled Root-LDPC ensemble are given by (3.22) – (3.26) for variable node types $1i$ and $1p$. DE equations for variable node types $2i$ and $2p$ can be derived by changing the node types and indices accordingly.*

- ***Local messages***

$$q_1^l = \epsilon_1 \left(1 - \left(1 - q_2^l\right)\left(1 - f_e f_1^l - g_e g_1^l\right)^{d_c-2}\left(1 - n_1^{l-1}\right)^{d_{cs}}\right)^{d_b-1}, \tag{3.22}$$

$$f_1^l = \epsilon_1 \left(1 - \left(1 - q_2^l\right)\left(1 - f_e f_1^l - g_e g_1^l\right)^{d_c-2}\left(1 - n_1^{l-1}\right)^{d_{cs}}\right)^{d_b-2} \left(1 - \left(1 - f_e f_2^l - g_e g_2^l\right)^{d_c-1}\left(1 - n_2^{l-1}\right)^{d_{cs}}\right), \tag{3.23}$$

$$g_1^l = \epsilon_1 (k_1^{l+1})^{d_{bs}} \left(1 - \left(1 - q_2^l\right)\left(1 - f_e f_1^l - g_e g_1^l\right)^{d_c-2}\left(1 - n_1^{l-1}\right)^{d_{cs}}\right)^{d_b-1}. \tag{3.24}$$

- ***Coupling messages (forward)***

$$n_1^l = \epsilon_1 (k_1^{l+1})^{d_{bs}-1} \left(1 - \left(1 - q_2^l\right) \left(1 - f_e f_1^l - g_e g_1^l\right)^{d_c-2} \left(1 - n_1^{l-1}\right)^{d_{cs}}\right)^{d_b}. \tag{3.25}$$

- ***Coupling messages (backward)***

$$k_1^{l+1} = 1 - \left(1 - q_2^{l+1}\right) \left(1 - f_e f_1^{l+1} - g_e g_1^{l+1}\right)^{d_c-1} \left(1 - n_1^l\right)^{d_{cs}-1}. \tag{3.26}$$

Proof. A reader who is familiar with the modern coding theory [86] can derive the DE equations from the local tree neighborhood of the nodes shown in Figure 3.10 – Figure 3.14. □

Since information bits are connected to rootchecks, the diversity order of the information bits is 2. To improve the coding gain of the information bits, we create more parity bits of diversity order 2 by using a Root-LDPC(2π) family instead of a Root-LDPC(4π) family (for more information about the Root-LDPC(2π) and Root-LDPC(4π) families refer to [16]).

Similar to Root-LDPC, the threshold boundary of coupled Root-LDPC is determined by the convergence of the densities associated with the local messages $1i(l) \to 2c(l)$ and $2i(l) \to 1c(l)$. In fact, we have L_c layered threshold boundaries corresponding to L_c copies of the Root-LDPC in the chain.

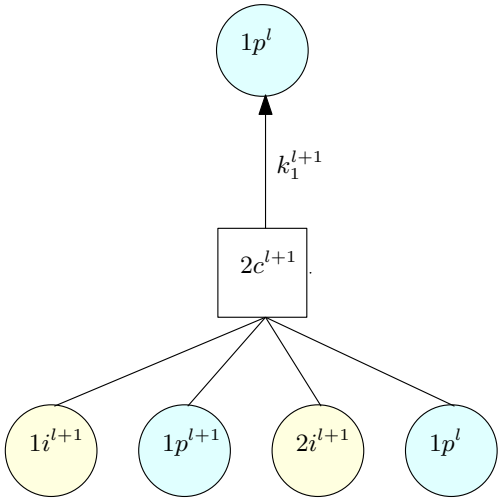


Figure 3.10: Local tree neighborhood of check node $2c$. This tree is used to determine the evolution of the backward coupling message k_1^{l+1}.

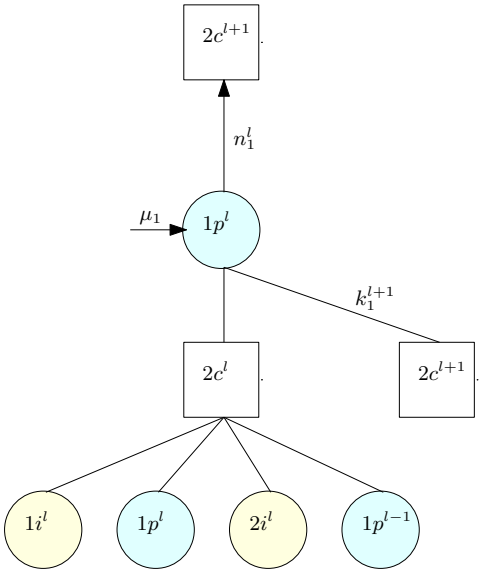


Figure 3.11: Local tree neighborhood of variable node $1p$. This tree is used to determine the evolution of the forward coupling message n_1^l.

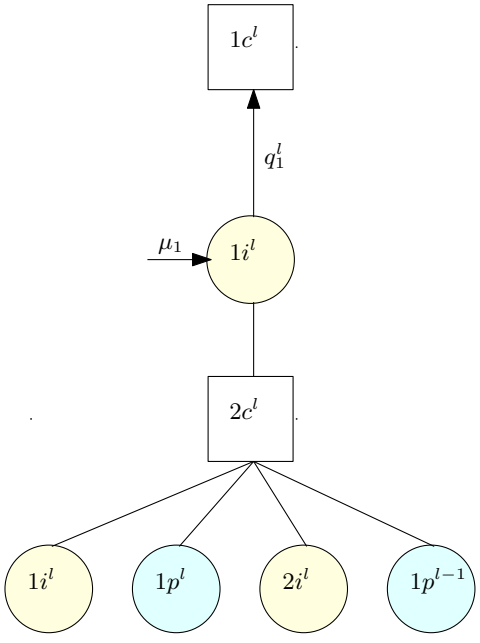


Figure 3.12: Local tree neighborhood of variable node $1i$. This tree is used to determine the evolution of the local message q_1^l.

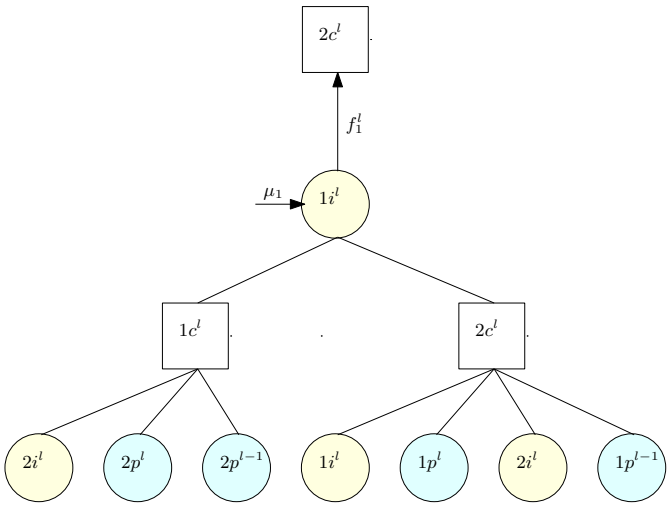


Figure 3.13: Local tree neighborhood of variable node $1i$. This tree is used to determine the evolution of the local message f_1^l.

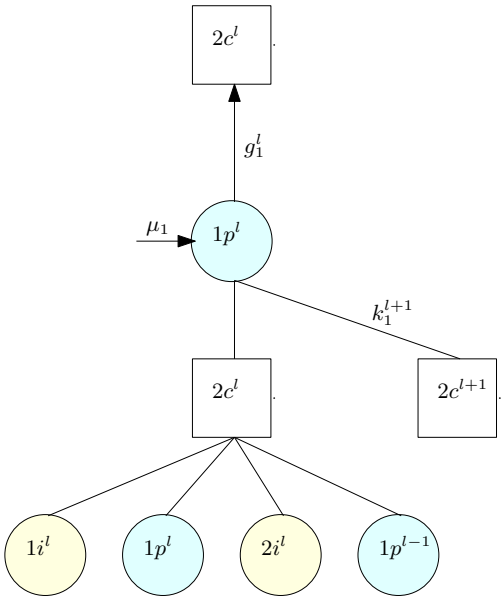


Figure 3.14: Local tree neighborhood of variable node $1p$. This tree is used to determine the evolution of the local message g_1^l.

Results

We present the outage boundaries of rate-1/2 random LDPC ensembles, uncoupled Root-LDPC ensembles, and coupled Root-LDPC ensembles in the erasure plane for two parallel erasure channels BEC(ϵ_1) and BEC(ϵ_2) shown in Figure 3.8.

The capacities of the erasure channels BEC(ϵ_1) and BEC(ϵ_2) are $C_1 = 1 - \epsilon_1$ and $C_2 = 1 - \epsilon_2$ respectively. When the two parallel channels are used uniformly, the capacity is the average of C_1 and C_2

$$C = \frac{1}{2}(C_1 + C_2) = 1 - \frac{1}{2}(\epsilon_1 + \epsilon_2). \tag{3.27}$$

For rate-1/2, the capacity limit is achieved when $C = R = 1/2$. Then, the capacity bound of the channel is given by the straight line $\epsilon_1 + \epsilon_2 = 1$.

Consider a random $(3, 6)$-LDPC ensemble. When $\epsilon_1 = \epsilon_2$, we have the standard DE equation for the $(3, 6)$-LDPC ensemble

$$x = \epsilon \left(1 - (1 - x)^5\right)^2, \tag{3.28}$$

and the BP threshold for $\epsilon_1 = \epsilon_2 = \epsilon$ is 0.429. Since the LDPC ensemble is random, a bit is transmitted through BEC(ϵ_1) or BEC(ϵ_2) equiprobably. So, the random LDPC is undergoing an average channel BEC($(\epsilon_1 + \epsilon_2)/2$). Then, the corresponding DE equation is

$$x = \frac{\epsilon_1 + \epsilon_2}{2} \left(1 - (1 - x)^5\right)^2, \tag{3.29}$$

and the ergodic threshold for the random $(3, 6)$-LDPC ensemble is given by the line $(\epsilon_1 + \epsilon_2)/2 = 0.429$. Similarly, the ergodic threshold for the random $(4, 8)$-LDPC ensemble is given by the line $(\epsilon_1 + \epsilon_2)/2 = 0.383$.

In the erasure plane, the (ϵ_1, ϵ_2) points under the threshold boundary cause the DE equations to converge and the (ϵ_1, ϵ_2) points above the threshold boundary result in the divergence of the DE equations. Thus, outage occurs when the (ϵ_1, ϵ_2) points are above the threshold boundary. The threshold boundary of the uncoupled $(3, 6)$ Root-LDPC is shown in Figure 3.15. The threshold boundary of the uncoupled $(3, 6)$ Root-LDPC is close to the ergodic threshold of the random $(3, 6)$-LDPC except for the cases where $\epsilon_1 = 0$ or $\epsilon_2 = 0$. As the uncoupled Root-LDPC has full diversity, $(\epsilon_1, \epsilon_2) = (1, 0)$ and $(\epsilon_1, \epsilon_2) = (0, 1)$ are on the threshold boundary of the uncoupled $(3, 6)$ Root-LDPC. However, the gap between the threshold boundary of the uncoupled $(3, 6)$ Root-LDPC and the capacity boundary of the channel is not small enough.

The threshold boundary improvement achieved by the proposed spatial coupling scheme is presented in Figure 3.16 for chain length $L_c = 60$. The chain length should be large enough, i.e. $L_c \geq 50$, so that the spatial coupling effect can be observed and the rate converges to the targeted rate. For larger values of L_c, the threshold boundary does not change. For simplicity, the parity variable nodes and check nodes are coupled with single edges, i.e. $d_{bs} = d_{cs} = 1$. Although we have L_c threshold boundaries for L_c copies of the Root-LDPC in the chain, the threshold boundaries are very close to each other. Thus, we plot a single boundary.

Figure 3.16 shows the threshold boundary of coupled (4,8) Root-LDPC. Although only a partial coupling scheme is applied by parity bits doping, the threshold boundary is very close to the capacity boundary of the channel. This significant threshold saturation is due to the coding gain introduced by spatial coupling. Figure 3.16 also shows that

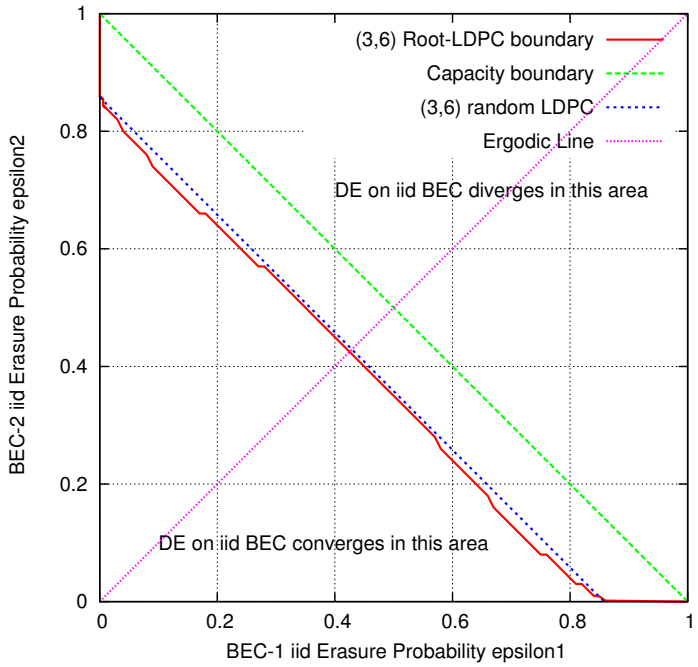


Figure 3.15: BEC threshold boundaries for random $(3, 6)$-LDPC and uncoupled $(3, 6)$ Root-LDPC ensembles in the erasure plane.

the coupled $(4, 8)$ Root-LDPC ensemble has full diversity. Note that, random rate-$1/2$ LDPC ensembles cannot achieve full diversity [16], which means that when all the bits transmitted through BEC(ϵ_1) are erased, it is not possible to recover these erased bits from the bits transmitted through BEC(ϵ_2) and vice versa. Thus, the ergodic threshold boundary lines for random LDPC ensembles do not pass through the points $(1, 0)$ and $(0, 1)$ in the erasure plane as shown and increasing the degrees of the variable nodes and check nodes deteriorates the threshold boundary of the random LDPC under BP decoding as presented in Figure 3.15 and Figure 3.16.

3.4 Spatial Coupling for Distributed Storage and Diversity Applications

In the context of distributed coding, both iterative decoding and locality aspects have motivated the interest in spatially-coupled LDPC codes that can achieve the maximum-a-

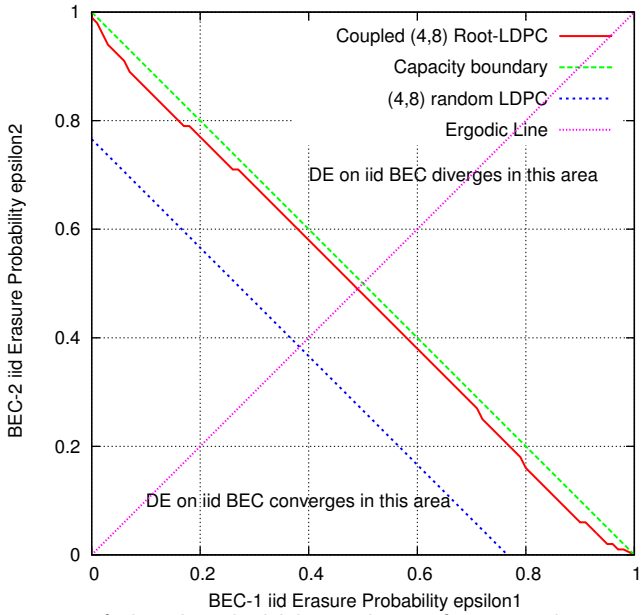


Figure 3.16: Saturation of the threshold boundary after coupling parity bits. Threshold boundaries for random $(4, 8)$-LDPC and coupled $(4, 8)$ Root-LDPC ensembles are shown in the erasure plane.

posteriori decoding threshold under iterative belief propagation. Moreover, Root-LDPC ensembles present some interesting properties. The Root-LDPC construction guarantees that information bits attain maximal diversity under iterative decoding over non-ergodic channels (e.g. block-fading and block-erasure channels). Notice that Root-LDPC codes allow only information bits to attain the targeted diversity while parity bits are usually diversity-deficient. Hence, we propose in this section to apply spatially-coupled Root-LDPC to distributed storage applications.

3.4.1 Channel coding model

The different coding parameters presented in sub-section 3.3.1 are now restated in the context of distributed storage. Assuming a storage domain with a total of ℓ machines partitioned into $M = 4$ clusters. Each cluster has its own erasure probability ϵ_i corresponding to one state of the 4-state block-erasure channel. Here, block-fading channels are replaced by block-erasure channels. Indeed, a block erasure is a special case of block fading (the two extremal fadings 0 and ∞). Satisfying diversity L in distributed coding means that the code is capable of filling all erasures if $L - 1$ clusters are erased. In

our construction, the data stored in one cluster may be totally erased, diversity order $L = 2$ ensures that all these erasures will be filled. The channel model is depicted in Figure 3.17, where N bits are transmitted on four parallel BECs with erasure probabilities ϵ_i. Chunks of $N/4$ bits of the Root-LDPC are transmitted on each BEC(ϵ_i), $i = 1 \dots 4$. Recall that for Root-LDPC codes, only the information bits are connected to rootchecks and have maximal diversity, leading to one cluster failures recovery. However, the coding gain of such code is weak due the lower diversity of parity bits. In order to improve the coding gain and saturate the threshold (outage) boundary, we propose to apply forward-layered spatially-coupled Root-LDPC ensembles presented in Section 3.3.2 to distributed storage. In the sequel, we shall use the coloring terminology to describe the four BEC channels with parameter ϵ_i: binary elements transmitted on BEC(ϵ_i) will be referred to as variable nodes of color i, $i = 1 \dots 4$.

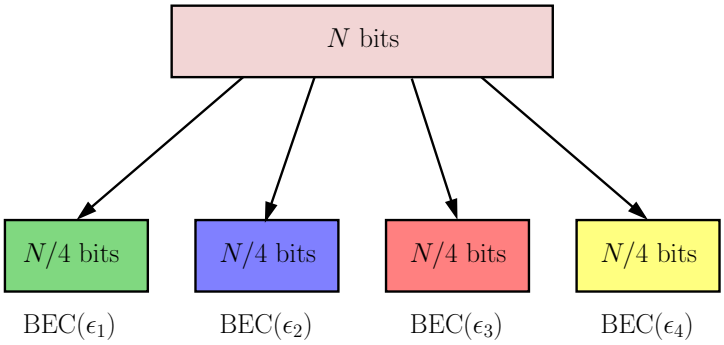


Figure 3.17: Parallel binary erasure channels where $N/4$ bits are transmitted on each channel BEC(ϵ_i), $i = 1 \dots 4$. N is the total code length.

3.4.2 Uncoupled rate-3/4 Root-LDPC ensemble

For the channel model described above, we split the first class of bits (green color) into information and parity bits respectively, i.e. $1i$ and $1p$. A similar splitting is made for all four colors leading to a total of eight classes of variable nodes as illustrated in Figure 3.18. Furthermore, four classes of check nodes are drawn on the right of the Tanner graph since the channel has $M = 4$ colors. Check nodes $1c$, $2c$, and $3c$ are rootchecks for class $1i$. Variable nodes $1p$ are connected to the non-root check node class $4c$. The number of edges for information and parity bits towards the non-root checks is Δ_i and Δ_p. The Root-LDPC structure is cyclic-symmetric. Similar edge connections are made for the three remaining colors. This ensemble shown in Figure 3.18 has rate 3/4. For each color, there are $3N/16$ information bits and $N/16$ parity bits. There are $N/16$ check nodes of a given class, all check nodes have degree $d_c = 3 + 3\Delta_i + \Delta_p$. Each information bit has node degree $1 + \Delta_i$ and each parity variable node has degree Δ_p. Then, the average variable-node degree d_b satisfies

$$d_b = \frac{3\Delta_i + \Delta_p + 3}{4} = \frac{d_c}{4}. \tag{3.30}$$

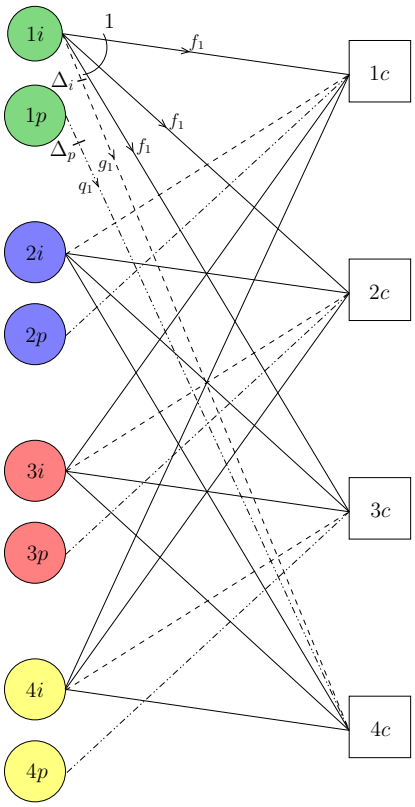


Figure 3.18: Compact Tanner graph representation for a rate-3/4 Root-LDPC ensemble for a channel with four colors.

The rootcheck structure guarantees double diversity in presence of block erasures. The proposed double-diversity maximal-rate code for four colors can also fill independent erasures. This capacity of handling both independent and burst erasures (per color) is studied now in the (ϵ_1, ϵ_2) plane. For simplicity, given the perfect cyclic-symmetry in the Root-LDPC ensemble, we assume that $\epsilon_2 = \epsilon_3 = \epsilon_4$ when drawing outage boundaries in the plane. The density evolution fixed-points are given below without any constraint on the four channel parameters. DE fixed-point equations can be found by drawing the local neighborhood of each type of variable nodes.

Let us define the message densities for the uncoupled rate-3/4 Root-LDPC ensemble as follows:

- f_1 is the density of message $1i \to 1c, 2c, 3c$.
- g_1 is the density of message $1i \to 4c$.
- q_1 is the density of message $1p \to 4c$.

Using the cyclic symmetry, messages f_i, g_i, and q_i are defined in a similar fashion for $i = 2, 3, 4$. Our LDPC ensemble corresponds to the following set of random graphs: edges

from $(1i,1p)$ towards $4c$ are constructed via a uniformly-selected socket permutation among the $(\Delta_i \frac{3N}{16} + \Delta_p \frac{N}{16})!$ permutations. The same construction is applied three more times for the three remaining colors. Hence, our rate-3/4 Root-LDPC ensemble is made out of four sets of interleavers. DE equations for the uncoupled ensemble are, for $i = 1 \dots 4$:

$$f_i = \epsilon_i \left(1 - (1 - f_e g_i - g_e q_i)^{3\Delta_i + \Delta_p - 1} \prod_{j \neq i} (1 - f_j) \right)^{\Delta_i}, \tag{3.31}$$

$$g_i = \epsilon_i \left(1 - (1 - f_e g_i - g_e q_i)^{3\Delta_i + \Delta_p - 1} \prod_{j \neq i} (1 - f_j) \right)^{\Delta_i - 1}$$
$$\cdot \frac{1}{3} \sum_{\substack{k=1 \\ k \neq i}}^{4} \left(1 - (1 - f_e g_k - g_e q_k)^{3\Delta_i + \Delta_p} \prod_{\substack{j \neq k \\ j \neq i}} (1 - f_j) \right), \tag{3.32}$$

$$q_i = \epsilon_i \left(1 - (1 - f_e g_i - g_e q_i)^{3\Delta_i + \Delta_p - 1} \prod_{j \neq i} (1 - f_j) \right)^{\Delta_p - 1}, \tag{3.33}$$

where the fractions f_e and g_e are defined as functions of the degrees between variable nodes and their non-root check nodes:

$$f_e = 1 - g_e = \frac{3\Delta_i}{3\Delta_i + \Delta_p}. \tag{3.34}$$

The (ϵ_1, ϵ_2) plane, restricted to $[0,1]^2$, is partitioned into two regions, the outage region $\mathcal{R}$ and the non-outage region $\overline{\mathcal{R}}$. The non-outage region is

$$\overline{\mathcal{R}} = \left\{ (\epsilon_1, \epsilon_2) \in [0,1]^2 \ : \ \mathbf{0} \text{ is the unique fixed-point} \right\}. \tag{3.35}$$

The boundary between $\mathcal{R}$ and $\overline{\mathcal{R}}$ will be called the *outage boundary* and denoted by $\mathcal{B}_o$. It can be compared to the capacity boundary found by writing that the capacity of the four BEC channels is equal to the coding rate, i.e. the capacity boundary is given by the line

$$\epsilon_1 + 3\epsilon_2 = 1. \tag{3.36}$$

The boundary $\mathcal{B}_o$ is shown in Figure 3.19 for the Root-LDPC ensemble with Tanner graph drawn in Figure 3.18. Double diversity in presence of block erasures (per color) is observed since $\mathcal{B}_o$ starts at the point $\epsilon_1 = 1$ and $\epsilon_2 = 0$. The BP threshold when all ϵ_i are equal is found on the so-called ergodic line ($\epsilon_1 = \epsilon_2$).

In the following, we propose a partial spatial coupling scheme for improving the boundary performance of the Root-LDPC ensemble.

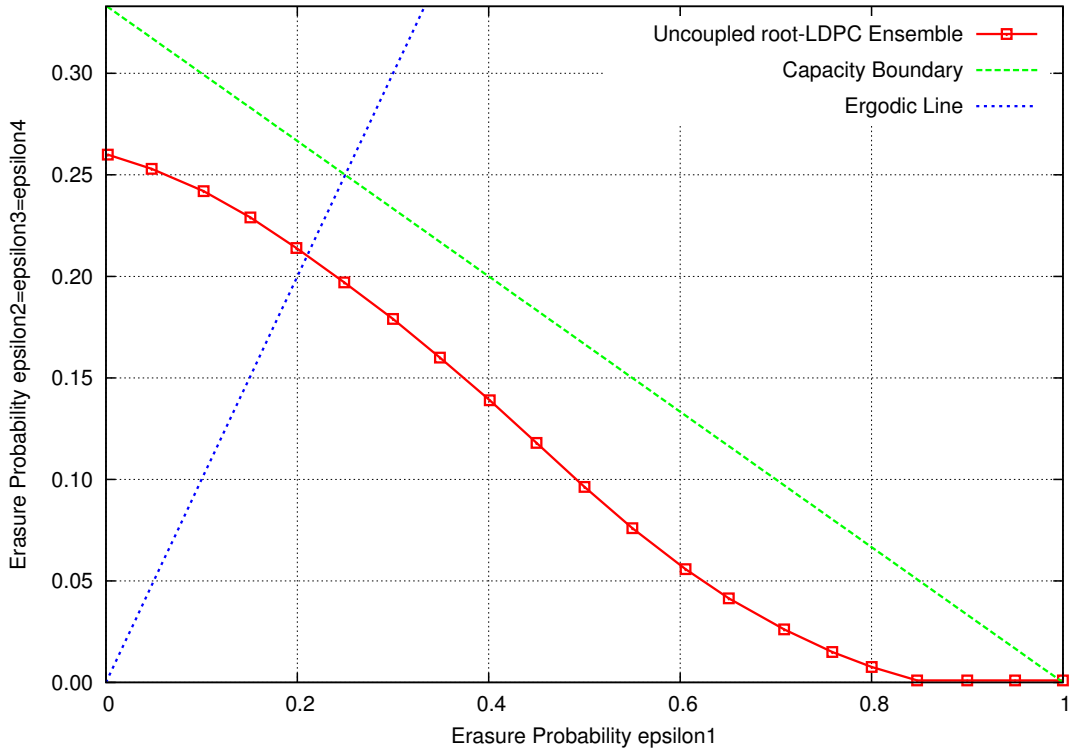


Figure 3.19: Outage boundary for rate-3/4 double-diversity Root-LDPC. Graph parameters are $\Delta_i = 2$ and $\Delta_p = 3$.

3.4.3 Double diversity Root-LDPC for distributed storage applications

Spatially-coupled Root-LDPC ensembles were introduced in Section 3.3 in the case of two-state non-ergodic channels ($M = 2$) with a coding rate 1/2. We build spatially-coupled Root-LDPC for four colors or equivalently four states ($M = 4$), where each state corresponds to a storage cluster.

The uncoupled ensemble of the previous sub-section is copied L_c times, each copy being placed on a spatial position ℓ, where $\ell = 1 \dots L_c$. In practice, the chain length L_c is taken to be large enough, e.g. $L_c = 100$. For the coupling scheme shown in Figure 3.20, the coupling window size is $w = 2$. Parity variable nodes at spatial position ℓ are coupled with non-root check nodes in spatial position $\ell + 1$. The chain is terminated at the right end with extra check nodes. For a general w, parity variable nodes at spatial position ℓ can be coupled with non-root check nodes in spatial positions $\ell + 1, \ell + 2, ..., \ell + (w - 1)$. The chain is terminated by adding check nodes at $w - 1$ spatial positions at the right end. The number of edges per variable node involved in coupling parity bits to the future $w - 1$ check nodes is Δ_c.

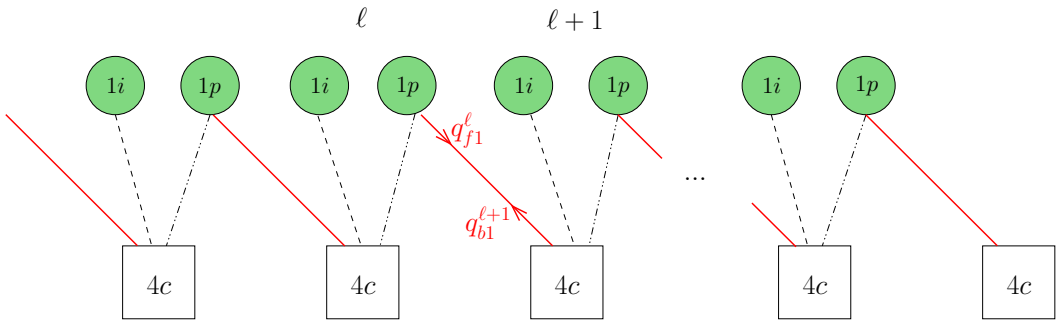


Figure 3.20: Spatial coupling structure for the Root-LDPC ensemble with a coupling window size $w = 2$. Only parity bits are coupled to check nodes in the next spatial position.

For spatial coupling with $w = 2$, the following messages are defined at spatial position ℓ:

- f_i^ℓ, g_i^ℓ, and q_i^ℓ are the same messages as in (3)-(5), used to represent evolution in the local ensemble at position ℓ, for the four colors $i = 1 \dots 4$.
- Messages f and g are not involved in spatial coupling, they are produced by information bits. For coupling parity bits, we keep the q notation. Message q_f stands for a forward message and q_b stands for a backward message. We define q_{fi}^ℓ as the left-to-right messages sent from variable nodes at position ℓ to check nodes at position $\ell + 1$. Similarly, q_{bi}^ℓ are backward messages from check nodes towards variable nodes in the previous spatial position, for $i = 1 \dots 4$.

Then, DE equations for the spatially coupled ($w = 2$) rate-3/4 Root-LDPC ensemble are

$$f_i^\ell = \epsilon_i \left(1 - (1 - f_e g_i^\ell - g_e q_i^\ell)^{3\Delta_i + \Delta_p - 1} \Pi_1\right)^{\Delta_i}, \tag{3.37}$$

$$g_i^\ell = \epsilon_i \left(1 - (1 - f_e g_i^\ell - g_e q_i^\ell)^{3\Delta_i + \Delta_p - 1} \Pi_1\right)^{\Delta_i - 1} \cdot \frac{1}{3} \sum_{\substack{k=1 \\ k \neq i}}^{4} \left(1 - (1 - f_e g_k^\ell - g_e q_k^\ell)^{3\Delta_i + \Delta_p} \Pi_3\right), \tag{3.38}$$

$$q_i^\ell = \epsilon_i \left(1 - (1 - f_e g_i^\ell - g_e q_i^\ell)^{3\Delta_i + \Delta_p - 1} \Pi_1\right)^{\Delta_p - 1} {q_{bi}^{\ell+1}}^{\Delta_c}, \tag{3.39}$$

$$q_{fi}^\ell = \epsilon_i \left(1 - (1 - f_e g_i^\ell - g_e q_i^\ell)^{3\Delta_i + \Delta_p - 1} \Pi_1\right)^{\Delta_p} {q_{bi}^{\ell+1}}^{\Delta_c - 1}, \tag{3.40}$$

$$q_{bi}^\ell = 1 - (1 - f_e g_i^\ell - g_e q_i^\ell)^{3\Delta_i + \Delta_p} \Pi_2, \tag{3.41}$$

where

$$\Pi_1 = \prod_{j\neq i} (1-f_j^\ell)(1-q_{fi}^{\ell-1})^{\Delta_c},\ \Pi_2 = \prod_{j\neq i} (1-f_j^\ell)(1-q_{fi}^{\ell-1})^{\Delta_c-1}, \text{and } \Pi_3 = \prod_{\substack{j\neq i\\ j\neq k}} (1-f_j^\ell)(1-q_{fk}^{\ell-1})^{\Delta_c}.$$

Density evolution fixed-points for $w > 2$ are given by equations similar to the five equations listed above. For $w = 3$, for $i = 1 \dots 4$, $q_{f_i}^\ell$ is splitted into:
• $q_{f_1 i}^\ell$ as the left-to-right messages sent from variable nodes at position ℓ to check nodes at position $\ell + 1$,
• $q_{f_2 i}^\ell$ as the left-to-right messages sent from variable nodes at position ℓ to check nodes at position $\ell + 2$.
Similarly, for $i = 1 \dots 4$, $q_{b_i}^\ell$ is splitted into:
• $q_{b_1 i}^\ell$ are backward messages from check nodes towards variable nodes in the previous spatial position,
• $q_{b_2 i}^\ell$ are backward messages from check nodes towards variable nodes in the $\ell-2$ spatial position.

The DE equations for a window size $w = 3$ are given by

$$f_i^\ell = \epsilon_i \left(1 - (1 - f_e g_i^\ell - g_e q_i^\ell)^{3\Delta_i+\Delta_p-1} \prod_{j\neq i} (1 - f_j^\ell)(1 - q_{f_1 i}^{\ell-1})^{\frac{\Delta_c}{2}} (1 - q_{f_2 i}^{\ell-2})^{\frac{\Delta_c}{2}}\right)^{\Delta_i} \tag{3.42}$$

$$g_i^\ell = \epsilon_i \left(1 - (1 - f_e g_i^\ell - g_e q_i^\ell)^{3\Delta_i+\Delta_p-1} \prod_{j\neq i} (1 - f_j^\ell)(1 - q_{f_1 i}^{\ell-1})^{\frac{\Delta_c}{2}} (1 - q_{f_2 i}^{\ell-2})^{\frac{\Delta_c}{2}}\right)^{\Delta_i-1} \times$$
$$\frac{1}{3} \sum_{\substack{k=1\\k\neq i}}^{4} \left(1 - (1 - f_e g_k^\ell - g_e q_k^\ell)^{3\Delta_i+\Delta_p} \prod_{\substack{j\neq k\\ j\neq i}} (1 - f_j^\ell)(1 - q_{f_1 i}^{\ell-1})^{\frac{\Delta_c}{2}} (1 - q_{f_2 i}^{\ell-2})^{\frac{\Delta_c}{2}}\right) \tag{3.43}$$

$$q_i^\ell = \epsilon_i \left(1 - (1 - f_e g_i^\ell - g_e q_i^\ell)^{3\Delta_i+\Delta_p-1} \prod_{j\neq i} (1 - f_j^\ell)(1 - q_{f_1 i}^{\ell-1})^{\frac{\Delta_c}{2}} (1 - q_{f_2 i}^{\ell-2})^{\frac{\Delta_c}{2}}\right)^{\Delta_p-1} \times$$
$$(1 - q_{b_1 i}^{\ell+1})^{\frac{\Delta_c}{2}} (1 - q_{b_2 i}^{\ell+2})^{\frac{\Delta_c}{2}} \tag{3.44}$$

$$q_{f_1 i}^\ell = \epsilon_i \left(1 - (1 - f_e g_i^\ell - g_e q_i^\ell)^{3\Delta_i+\Delta_p-1} \prod_{j\neq i} (1 - f_j^\ell)(1 - q_{f_1 i}^{\ell-1})^{\frac{\Delta_c}{2}} (1 - q_{f_2 i}^{\ell-2})^{\frac{\Delta_c}{2}}\right)^{\Delta_p} \times$$
$$(1 - q_{b_1 i}^{\ell+1})^{\frac{\Delta_c}{2}-1} (1 - q_{b_2 i}^{\ell+2})^{\frac{\Delta_c}{2}} \tag{3.45}$$

$$q_{f_2 i}^{\ell} = \epsilon_i \left(1 - (1 - f_e g_i^{\ell} - g_e q_i^{\ell})^{3\Delta_i + \Delta_p - 1} \prod_{j \neq i} (1 - f_j^{\ell})(1 - q_{f_1 i}^{\ell-1})^{\frac{\Delta_c}{2}} (1 - q_{f_2 i}^{\ell-2})^{\frac{\Delta_c}{2}} \right)^{\Delta_p} \times$$
$$(1 - q_{b_1 i}^{\ell+1})^{\frac{\Delta_c}{2}} (1 - q_{b_2 i}^{\ell+2})^{\frac{\Delta_c}{2} - 1} \tag{3.46}$$

$$q_{b_1 i}^{\ell} = 1 - (1 - f_e g_i^{\ell} - g_e q_i^{\ell})^{3\Delta_i + \Delta_p} \prod_{j \neq i} (1 - f_j^{\ell})(1 - q_{f_1 i}^{\ell-1})^{\frac{\Delta_c}{2} - 1} (1 - q_{f_2 i}^{\ell-2})^{\frac{\Delta_c}{2}} \tag{3.47}$$

$$q_{b_2 i}^{\ell} = 1 - (1 - f_e g_i^{\ell} - g_e q_i^{\ell})^{3\Delta_i + \Delta_p} \prod_{j \neq i} (1 - f_j^{\ell})(1 - q_{f_1 i}^{\ell-1})^{\frac{\Delta_c}{2}} (1 - q_{f_2 i}^{\ell-2})^{\frac{\Delta_c}{2} - 1} \tag{3.48}$$

The outage boundary for the coupled Root-LDPC is shown in Figure 3.21 for a window size $w = 2$, 3, and 5. The new outage boundary approaches the capacity line $\epsilon_1 + 3\epsilon_2 = 1$. Nevertheless, we do not observe a boundary saturation as it is expected in spatially coupled LDPC codes. The reasons are mainly due to the partial spatial coupling of parity bits only.

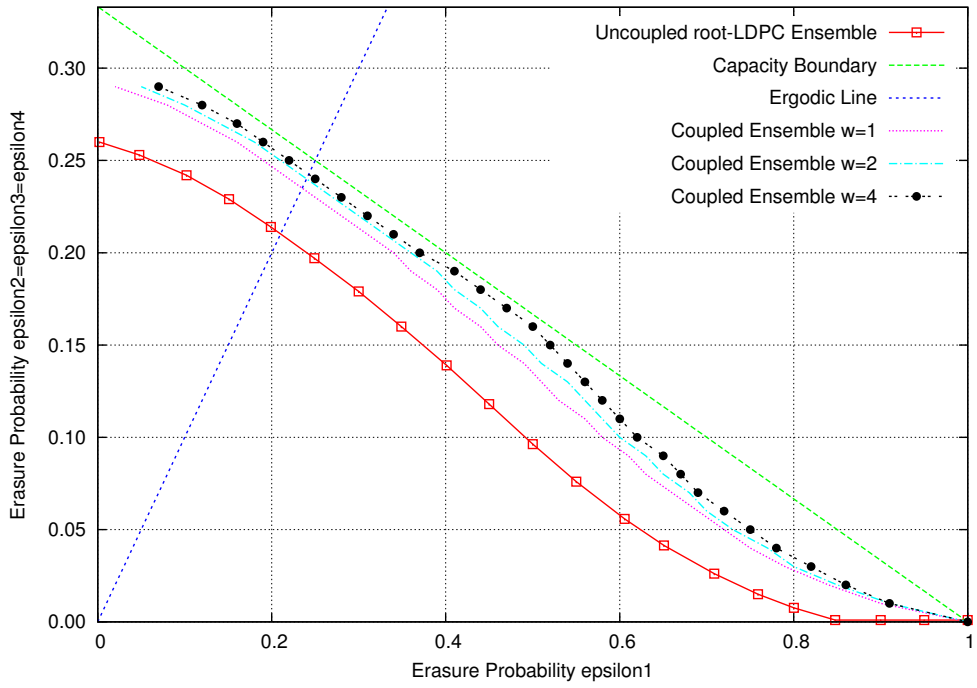


Figure 3.21: Boundary performance for rate-3/4 double-diversity Root-LDPC with spatial coupling of window size $w = 2, 3, 5$. The graph parameters are $\Delta_c = w - 1$, $\Delta_i = 2$, and $\Delta_p = 3, 5, 10$.

3.5 Conclusions

In this Chapter, a new method called forward layered coupling, for spatial coupling of low-density parity-check ensembles was proposed. The method is inspired from overlapped layered coding and has the shortest possible memory, i.e. $w = 2$. Edges of local ensembles and those defining the spatial coupling are separately built. Thresholds for regular coupled chains were given in Table 3.1. The new method also allows the construction of non-uniform coupling chains with near-Shannon spatially-varying thresholds under iterative decoding.

The second part of this chapter was dedicated to present novel forward-layered coupling schemes for Root-LDPC. Local protographs are coupled by new edges connected to their parity bits only. As expected from parity bits doping [16], coupling parity bits is greatly enhancing the coding gain of information bits. The spatially-coupled Root-LDPC ensembles achieve threshold boundaries very close to the capacity boundary (saturation) in the erasure plane. This behavior in the erasure plane will automatically result in a saturation of the Root-LDPC outage boundary on a real block-fading channel with additive noise.

In the last section, spatially-coupled Root-LDPC are applied to storage and diversity applications. From Definition 2.1 and the structure of the double-diversity Root-LDPC codes, it is obvious that information bits have locality $d_c - 1$, i.e. the degree of check nodes minus one edge, in the coupling scheme presented in Section 3.4.3. In fact, this is true for any LDPC code when considering independent erasures. In the special case where erasures occur per color (block erasures) the locality of a random LDPC ensemble is $O(\log n)$ whereas that of a Root-LDPC is maintained at $d_c - 1$ thanks to rootcheck nodes of order 1. Usually, we would like to maintain all information bits at root order 1. This forces the construction to couple parity bits only. We also saw that the outage boundary is not completely saturated towards the capacity line. Root order 1 for all information bits is too constraining for rate 3/4 and 4 colors. A potential solution to saturate the boundary of coupled Root-LDPC (as a perspective for future work) is to introduce a fraction of order 2 information bits. This should allow the enhancement of spatial coupling by engaging both information and parity bits.

In this chapter we dealt with spatially-coupled LDPC ensembles and spatially-coupled Root-LDPC ensembles. These codes achieve a good compromise between obtaining good locality and iterative decoding process. However, LDPC and Root-LDPC are not suitable for distributive storage applications requiring data updating. Indeed, updating the data is equivalent to encoding the data in term of complexity. As a result of the sparsity of LDPC code parity-check matrix, the generator matrix is dense. Consequently, data encoding and updating are heavy processes. In the next chapter a new coding scheme using double-diversity product code is proposed to overcome this problem.

Chapter 4

Edge Coloring in Product Code

The colossal amount of data stored or conveyed by network nodes requires a special design of coding structures to protect information against loss or errors and to facilitate its access. At the end-user level, coding is essential for transmitting information towards the network whether it is located in a single node or distributed over many nodes. At the network level, coding should help nodes to reliably save a big amount of data and to efficiently communicate with each others. Powerful capacity-achieving error-correcting codes developed in the last two decades are mainly efficient at large or asymptotic block length, e.g. low-density parity-check codes (LDPC) [36] and their spatially-coupled ensembles [56], parallel-concatenated convolutional (Turbo) codes [8][6], and polar codes derived from channel polarization [5]. Data transmission and storage in many nowadays networks may require short-length packets that are not suitable for capacity-achieving codes. The current interest in finite-length channel coding rates [74] put back the light on code design for short and moderate block length. Many potential candidates are available for this non-asymptotic length context such as binary and non-binary BCH codes, including Reed-Solomon (RS) codes, Reed-Muller (RM) codes, and tensor product codes of all these linear block codes [65][10][61].

Product codes, introduced by Peter Elias in 1954 [31], are tensor products of two (or more) simple codes with a structure that is well-suited to iterative decoding via its graphical description. In the early decades after their invention, product codes received a great attention due to their capability of correcting multiple burst errors [117][109], the availability of erasure-error bounded-distance decoding algorithms [113], the ability of correcting many errors beyond the guaranteed correction capacity [1], and their efficient implementation with a variable rate [115]. The pioneering work by Tanner [101] brought new tools to coding theory and put codes on graphs, including product codes, and their iterative decoding in the heart of modern coding theory [53][52][86]. The graph approach of coding led to new optimal cycle codes on Ramanujan/Cayley graphs [104] and to Generalizations of LDPC and product codes, known as GLD codes, studied for the binary symmetric channel (BSC) and the Gaussian channel [13]. The excellent performance of iterative (turbo) decoding of product codes on the Gaussian channel [76] made them compete with Turbo codes and LDPC codes for short and moderate block length. The

convergence rate and stability of product codes iterative decoding were studied based on a geometric framework [94]. Product codes with mixed convolutional and block components were also found efficient in presence of impulsive noise [35]. In addition, iterated Reed-Muller product codes were shown to exhibit good decoding thresholds for the binary erasure channel, but at high and low coding rates only [110].

The class of product codes in which the row and the column code are both Reed-Solomon codes was extensively used since more than two decades in DVD storage media and in mobile cellular networks [116]. In these systems, the channel is modeled as a symbol-error channel without soft information, i.e. suited to algebraic decoding. Improvements were suggested for these RS-based product codes such as soft information provided by list decoding [90] within the iterative process in a Reddy-Robinson framework [82]. Also, RS-based product codes were directly decoded via a Guruswami-Sudan list decoder [43] after being generalized to bivariate polynomials [4]. For general tensor products of codes and interleaved, a recent efficient list decoding algorithm was published [39], with an improved list size in the binary case. On channels with soft information, RS-based product codes may be row-column decoded with soft-decision constituent decoders [32][48].

Tolhuizen found the Hamming weight distribution of both binary and non-binary product codes up to a weight less than $d_1 d_2 + \max(d_1 \lceil d_2/q \rceil, d_2 \lceil d_1/q \rceil)$ [105]. Enumeration of erasure patterns up to a weight less than $d_1 d_2 + \min(d_1, d_2)$ was realized by Sendrier for product codes with MDS components [95]. Rosnes studied stopping sets of binary product codes under iterative ML-component-wise decoding [88], where the defined stopping sets and their analysis are based on the generalized Hamming distance [114][44].

In this chapter, we consider non-binary product codes with MDS components and their iterative algebraic decoding on the erasure channel. Both independent and block erasures are considered. The erasure channel is currently a major area of research in coding theory [57][58] because of strong connections with theoretical computer science [58] and its model that easily allows to understand the behavior of codes such as for LDPC codes [27], for general linear block codes [93], and for turbo codes [87]. Coding for block erasures was examined by Lapidoth in the context of convolutional codes [59]. This was a basis to later construct codes for the block-fading channel with additive white Gaussian noise [42][17]. The notion of *rootcheck* introduced in [17][16] for single-parity check nodes was applied to more general check nodes in GLD codes [15] and product codes [14] to achieve diversity on non-ergodic block-fading channels. The rootcheck concept is the main tool in this chapter, in a way similar to [14], to define a compact graph representation and study iterative decoding in presence of block erasures. Edge coloring is one of the most interesting problems in modern graph theory [11]. In this chapter, edge coloring is a tool, when combined to the rootcheck concept, yields double-diversity product codes. Our work is valid for finite-length MDS-based product codes only. Product codes for asymptotic block length were studied for single-parity codes constituents [77] and for the erasure channel with a standard regular structure [92] and

MDS-based irregular structures [2].

Whether a product code is endowed with an edge coloring or not, the analysis of stopping sets, their characterization and their enumeration is a fundamental task to be able to design codes for erasure channels and determine the decoder performance. Our work in this sense is an improvement to previous works cited above by Tolhuizen, Sendrier, and Rosnes. Besides this objective of stopping sets characterization which is useful for independent channel erasures and erasures occurring in blocks of symbols, recent works on locality [40] stimulated us to search for edge colorings with a large population of edges that admit a minimal rootcheck order. Locality is a concept encountered in distributive storage [54][80] where classic coding theory is adapted to the nature of a network with distributed nodes with its own constraints of load in bandwidth and storage [29][70]. Furthermore, product codes with MDS components appear to be suited to distributive storage [33] owing to their simple and mature techniques of erasure resilience. In our search for good edge colorings, we provide a new algorithm based on the concept of differential evolution [99][71]. We use no crossover in our evolution loop, only a mutation of the population of bad edges is made to search for a better edge coloring. Our MDS-based product codes equipped with a double-diversity edge coloring are suited to distributed storage applications and to wireless networks where diversity is a key parameter.

The chapter is structured as follows. Section 4.1 gives a list of mathematical notations. The graph representation of product codes is given in Section 4.2, including compact and non-compact graphs. Also the rootcheck concept and its consequences are also found in Section 4.2. The analysis of stopping sets is made in Section 4.3. Our edge coloring algorithm for product codes is described in Section 4.4. Finally, in Section 4.5, we study the performance of product codes with MDS components on erasure channels and we give theoretical and numerical results before the conclusions in the last section.

4.1 Mathematical notation and Terminology

We start by the notation related to the product code and its row and column components. The impatient reader may skip this entire section and then refer to it later to clarify any notation within the text. Basic notions on product codes and fundamental properties are found in main textbooks [65][10][61] and the encyclopedia of telecommunications [52].
The column code C_1 is a linear block code over the finite field $\mathbb{F}_q$ with parameters $[n_1, k_1, d_1]_q$ which may be summarized by $[n_1, k_1]$ when no confusion is possible. The integer q is the code alphabet size, n_1 is the code length, k_1 is the code dimension as a vector subspace of $\mathbb{F}_q^{n_1}$, and d_1 is the minimum Hamming distance of C_1. Similarly, the row code C_2 is a linear block code with parameters $[n_2, k_2, d_2]_q$. Let G_1 and G_2 be two matrices of size $k_1 \times n_1$ and $k_2 \times n_2$ containing in their row a basis for the subspaces C_1 and C_2 respectively. From the two generator matrices G_1 and G_2 a product code C_P is constructed as a subspace of $\mathbb{F}_q^N$ with a generator matrix $G_P = G_1 \otimes G_2$, where $N = n_1 n_2$ and $\otimes$ denotes the Kronecker product [65]. C_P has dimension $K = k_1 k_2$ and minimum Hamming distance $d_P = d_1 d_2$. C_1 and C_2 are also called component codes,

this is a terminology from concatenated codes. In [101] and [14], vertices associated to component codes are called subcode nodes.

A linear $[n, k, d]_q$ code is said to be MDS, i.e. Maximum Distance Separable, if it satisfies $d = n - k + 1$. Binary MDS codes are the trivial repetition codes and the single parity-check codes. In this chapter, we only consider non-trivial non-binary MDS codes where $q > n > 2$. A linear code over $\mathbb{F}_q$ of rate $R = k/n$ is said to be MDS diversity-wise or MDS in the block-fading/block-erasure sense if it achieves a diversity order L such that $L = 1 + \lfloor M(1-R) \rfloor$, where M is the number of degrees of freedom in the channel. The right term $1 + \lfloor M(1-R) \rfloor$ is known as the block-fading Singleton bound [66][51]. In this chapter, M shall denote the number of colors, i.e. the palette size of an edge coloring. Assume that code symbols are partitioned into M sub-blocks, a code is said to attain diversity L if it is capable of correct decoding when $L-1$ sub-blocks are erased by the channel. The reader should refer to [108], chapter 3, for an exact definition of diversity on fading channels with additive white Gaussian noise.

A product code shall be represented by a non-compact graph $\mathcal{G} = (V_1, V_2, E)$. $\mathcal{G}$ is a complete bipartite graph where V_1 is the set of n_2 right vertices, V_2 is the set of n_1 left vertices, and E is the set of N edges representing the code symbols. A compact graph $\mathcal{G}^c$ will also be introduced in the next section with $\mathcal{G}^c = (V_1^c, V_2^c, E^c)$. The number of edges (also called super-edges) in the compact graph is $|E^c| = N^c$. A super-edge is equivalent to a super-symbol that represents $(n_1 - k_1)(n_2 - k_2)$ symbols from $\mathbb{F}_q$. The ensemble of edge colorings is denoted $\Phi(E)$ and $\Phi(E^c)$ for $\mathcal{G}$ and $\mathcal{G}^c$ respectively. An edge coloring will be denoted by ϕ. Given ϕ, the rootcheck order of an edge is $\rho(e)$. The greatest $\rho(e)$ among all edges will be referred to as $\rho_{max}(\phi)$. The number of edges e satisfying $\rho(e) = 1$ is $\eta(\phi)$, this is the number of good edges and will be processed by the DECA algorithm in Section 4.4. The DECA parameter $\aleph$ shall represent the number of edges to be mutated, i.e. those edges being chosen in the population of bad edges satisfying $\rho(e) > 1$.

Under iterative row-column decoding, the rootcheck order ρ is equal to the number of decoding iterations required to solve the edge value (or the symbol associated to that edge). In this chapter, one decoding iteration is equivalent to decoding all rows or decoding all columns. A sequence of n_1 row decoders followed by a sequence of n_2 column decoders is counted as two decoding iterations.

We give now a general definition of a stopping set. A detailed study is found in Section 4.3. The notion of a stopping set is useful for iterative decoding in presence of erasures [27].

Definition 4.1. Let $C[n, k]_q$ be a linear code. Assume that the symbols of a codeword are transmitted on an erasure channel. The decoder $\mathcal{D}$ is using some deterministic decoding method. Consider a set $\mathcal{S}$ of s fixed positions $i_1, i_2, \ldots, i_s$ where $1 \leq i_j \leq n$. The set $\mathcal{S}$ is said to be a Stopping Set if $\mathcal{D}$ fails in retrieving the transmitted codeword when all symbols on the s positions given by $\mathcal{S}$ are erased.

This chapter focuses on stopping sets of a product code under iterative algebraic row-column decoding, i.e. referred to as type II stopping sets. The number of stopping

sets of size w is τ_w. The rectangular support $\mathcal{R}(\mathcal{S})$ of a stopping set $\mathcal{S}$ can be seen as the smallest rectangle containing $\mathcal{S}$. After excluding rows and columns not involved in $\mathcal{S}$, the rectangular support has size $\ell_1 \times \ell_2$ where $w = |\mathcal{S}| \leq \ell_1 \ell_2$. The word error performance of C_P shall be estimated on erasure channels, P_{ew}^{ML} is the word error probability under Maximum Likelihood decoding and $P_{ew}^{\mathcal{G}}$ is the word error probability under iterative row-column decoding. Three erasure channels are considered: 1- The Symbol Erasure Channel, $SEC(q, \epsilon)$, where code symbols are independently erased with a probability ϵ, 2- The Color Erasure Channel, $CEC(q, \epsilon)$, where all symbols associated to the same color are block-erased with a probability ϵ. On the $CEC(q, \epsilon)$, block-erasure events are independent from one color to another. 3- The unequal probability Symbol Erasure Channel, $SEC(q, \{\epsilon_i\}_{i=1}^{M})$, where symbol erasures are independent but their erasure probability varies from one color to another.

4.2 Graph representations for diversity

Efficient graph representation of codes was established by Tanner for different types of coding structures [101]. Bounds on the code parameters and iterative decoding algorithms were also proposed for codes on graphs [101]. In this chapter, we study the edge coloring of a product code graph, where edges represent code symbols. As shown below, the original graph for a product code is too complex, i.e. it leads to a large ensemble of colorings. Hence, we introduce a compact graph where symbols are grouped together with the same color in order to reduce the size of the coloring ensemble. The compact graph also has another asset: grouping parity symbols together renders check nodes similar to parity-check nodes found in standard low-density parity-check codes [36] [86].

4.2.1 Non-compact graph

Consider a product code $C_1[n_1, k_1]_q \otimes C_2[n_2, k_2]_q$ where C_1 is the column code and C_2 is the row code. The product code is defined over the finite field $\mathbb{F}_q$ and has length N and dimension K given by [65]

$$N = n_1 n_2, \qquad K = k_1 k_2. \tag{4.1}$$

Each code symbol simultaneously belongs to one row and to one column. Product codes studied in this thesis report are regular, in the sense that all columns are codewords of C_1 and all rows are codewords of C_2. The graph of $C_1[n_1, k_1]_q \otimes C_2[n_2, k_2]_q$ is built as follows. We use the same terminology as in [86]:

- n_1 check nodes are drawn on the left. A left check node represents the coding constraint which states that a row belongs to C_2. The n_1 left check nodes are referred to as C_2 check nodes, or row check nodes, or equivalently left vertices.
- n_2 check nodes are drawn on the right. A right check node represents the coding constraint which states that a column belongs to C_1. The n_2 right check nodes are referred to as C_1 check nodes, or column check nodes, or equivalently right vertices.

- An edge is drawn between a left vertex and right vertex. It represents a code symbol located on the row of the left vertex and on the column of the right vertex. The code symbol belongs to $\mathbb{F}_q$.

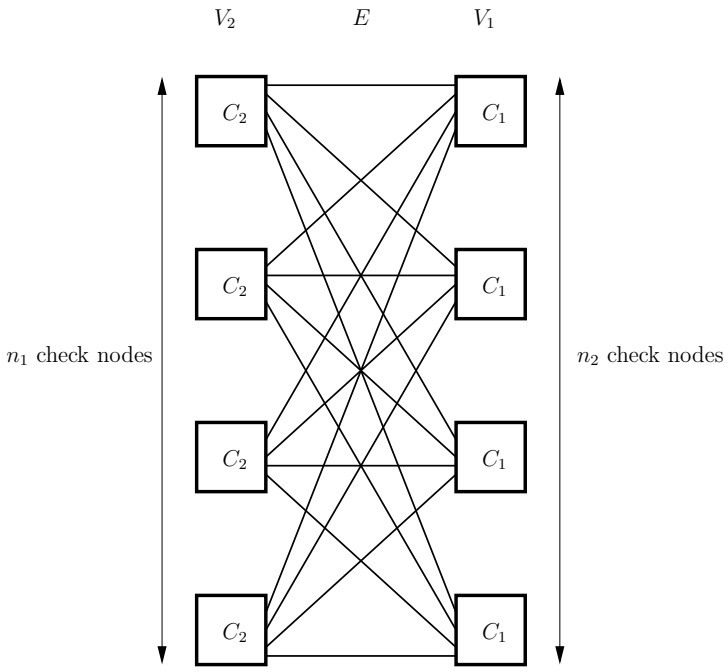


Figure 4.1: Non-compact bipartite graph $\mathcal{G} = (V_1, V_2, E)$ of a product code $[4, 2]^{\otimes 2}$, i.e. $n_1 = n_2 = 4$, $k_1 = k_2 = 2$, $|V_1| = |V_2| = 4$, and $|E| = N = n_1 n_2 = 16$ edges representing 16 symbols in $\mathbb{F}_q$.

In summary, the product code graph (V_1, V_2, E) is a complete biregular bipartite graph built from n_1 left vertices, n_2 right vertices, and $N = |E| = n_1 n_2$ edges representing code symbols. The left degree is n_2 and the right degree is n_1. Irregular product codes can be found in [2]. Our chapter is restricted to regular product codes. Figure 4.1 shows the bipartite graph of a square regular symmetric product code $[4, 2] \otimes [4, 2]$. The graph structure reveals n_1, n_2, and $N = n_1 n_2$. The dimensions k_1 and k_2 of the component codes have no effect on the number of vertices and edges in the product code graph. Indeed, a $[4, 3] \otimes [4, 3]$ code can also be defined by the graph in Figure 4.1. The role of the dimensions k_1 and k_2 is played within the check constraints inside left and right vertices. Similarly, the size of the finite field defining the code cannot be revealed from the graph structure, i.e. the product code graph does not depend on q.

Definition 4.2. The non-compact graph $\mathcal{G} = (V_1, V_2, E)$ for a $[n_1, k_1] \otimes [n_2, k_2]$ product code is a complete bipartite graph with $n_1 = |V_2|$ left vertices and $n_2 = |V_1|$ right vertices.

4.2.2 Compact graph

In [14] where the diversity of binary product codes was considered, vertices of the non-compact graph were grouped together into super-vertices (or supernodes) because the different channel states lead to multiple classes of check nodes as in root-LDPC codes [17]. To render a graph-encodable code, supernodes in [14] were made by putting $n-k$ nodes together for a $[n,k]$ component code. Also, $n-k$ is not necessarily a divisor of n.

Definition 4.3. The compact graph $\mathcal{G}^c = (V_1^c, V_2^c, E^c)$ for a $[n_1,k_1] \otimes [n_2,k_2]$ product code is a complete bipartite graph with $\lceil \frac{n_1}{n_1-k_1} \rceil = |V_2^c|$ left vertices and $\lceil \frac{n_2}{n_2-k_2} \rceil = |V_1^c|$ right vertices.

From the above definition, the number of edges in the compact graph $\mathcal{G}^c$ is found to be

$$N^c = |E^c| = \left\lceil \frac{n_1}{n_1-k_1} \right\rceil \times \left\lceil \frac{n_2}{n_2-k_2} \right\rceil . \tag{4.2}$$

Assuming that (n_1-k_1) divides n_1 and and (n_2-k_2) divides n_2, a left check node in $\mathcal{G}^c$ is equivalent to n_2-k_2 row constraints and a right check node in $\mathcal{G}^c$ is equivalent to n_1-k_1 column constraints. A edge in the compact graph carries $(n_1-k_1)\times(n_2-k_2)$ code symbols. To avoid confusion between edges of $\mathcal{G}$ and $\mathcal{G}^c$, we may refer to those in $\mathcal{G}^c$ as super-edges or equivalently as super-symbols. If n_i is not multiple of n_i-k_i, then the last row or column supernode will contain less than n_i-k_i check nodes. Figure 4.2 depicts the compact graph of the $[4,2]^{\otimes 2}$ product code. All $[n,n/2]^{\otimes 2}$ product codes have a compact graph identical to that of $[4,2]^{\otimes 2}$, for all $n \geq 2$, n even.

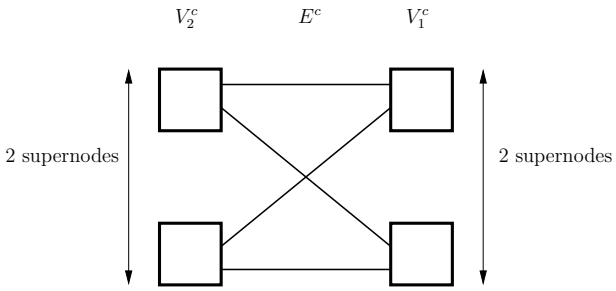


Figure 4.2: Compact bipartite graph $\mathcal{G}^c = (V_1^c, V_2^c, E^c)$ with two supernodes on each side for the product code $[n,n/2]^{\otimes 2}$, $|V_1^c| = |V_2^c| = 2$ and $|E^c| = N^c = 4$ supersymbols. Each super-symbol (i.e. super-edge) contains $n^2/4$ symbols (i.e. edges).

4.2.3 Diversity and codes on graphs

From a coding point of view, diversity is the art of creating many replicas of the same information. From a channel point of view, diversity is the number of degrees of freedom

available while transmitting information. In distributive storage, independent failure of individual machines is modeled by independent erasures of code symbols, while the outage of a cluster of machines is modeled as block erasures of code symbols. Assuming a storage domain with a large set of machines partitioned into M clusters, diversity of distributed coding is defined as follows:

Definition 4.4. Consider a product code C_P defined over $\mathbb{F}_q$. Assume that symbols are given M different colors. Erasing one color is equivalent to erasing all symbols having this color. The code is said to achieve a diversity L if it is capable of filling all erasures after erasing $L-1$ colors. The code is full-diversity when $L = M$.

The integer L may also be called the diversity order. For Gaussian channels with fading, the diversity order appears as the slope of the error probability, i.e. $L = \lim_{\gamma\to\infty} -\frac{\log P_e}{\log \gamma}$ [17]. In the above definition, a cluster has been replaced by a color. We will use this terminology throughout the chapter. Notice that coloring symbols is equivalent to edge coloring of the product code graph. The number of edges is N in the non-compact graph and N^c in the compact graph. In the sequel, all colorings are supposed to be perfectly balanced, i.e. M divides both N and N^c and the number of edges having the same color is N/M and N^c/M for the non-compact graph and the compact graph respectively. More formally, our edge coloring is defined as follows: an edge coloring ϕ of $\mathcal{G} = (V_1, V_2, E)$ is a mapping associating one color to every edge in E,

$$\phi : E \to \{1, 2, \ldots, M\}, \tag{4.3}$$

such that $|\phi^{-1}(i)| = N/M$ for $i = 1 \ldots M$, where $\phi^{-1}(i)$ is the inverse image of i. Similarly, $\phi : E^c \to \{1, 2, \ldots, M\}$ for $\mathcal{G}^c = (V_1^c, V_2^c, E^c)$ and $|\phi^{-1}(i)| = N^c/M$. The set of such mappings for $\mathcal{G}$ and $\mathcal{G}^c$ is denoted $\Phi(E)$ and $\Phi(E^c)$ respectively.
Consider a coloring ϕ in $\Phi(E^c)$. It can be embedded into $\Phi(E)$ by copying the color of a super-edge to its associated $(n_1 - k_1) \times (n_2 - k_2)$ edges in E. Thus, let $\Phi(E^c \to E)$ be the subset of colorings in $\Phi(E)$ obtained by embedding all colorings of $\Phi(E^c)$ into $\Phi(E)$. We have

$$\Phi(E^c \to E) \subset \Phi(E) \quad \text{and} \quad |\Phi(E^c \to E)| = |\Phi(E^c)|. \tag{4.4}$$

The size of the edge coloring ensembles $\Phi(E)$ and $\Phi(E^c)$ is obviously not the same when $N^c < N$, which occurs for both row and column component codes not equal to single parity-check codes. Indeed, when a palette of size M is used to color edges, the total number of colorings of E is

$$|\Phi(E)| = \frac{N!}{((N/M)!)^M}. \tag{4.5}$$

This number for the compact graph is

$$|\Phi(E^c)| = \frac{N^c!}{((N^c/M)!)^M}. \tag{4.6}$$

As an example, for the $[12, 10]^{\otimes 2}$ code and $M = 4$, there are $2 \cdot 10^{83}$ edge colorings for the non-compact graph and $2 \cdot 10^{19}$ edge colorings for the compact graph. It is clear that the construction of product codes for diversity is much easier when based on $\mathcal{G}^c = (V_1^c, V_2^c, E^c)$ because its edge coloring ensemble is smaller. Furthermore, as described below, vertices in $\mathcal{G}^c$ act in a way similar to standard LDPC check nodes making the design very simple. Furthermore, we will see in Section 4.3 that edge colorings of the compact graph render larger stopping sets than colorings of the non-compact graph.

The diversity order L attained by a code can never exceed M, the latter being the diversity from a channel point of view. A tighter upper bound of L showing the rate-diversity tradeoff is the block-fading Singleton bound. The Singleton bound for the maximal achievable diversity order is valid for all types of non-ergodic channels, including block-erasure and block-fading channels. The block-fading Singleton bound states that [51] [66]

$$L \leq 1 + \lfloor M(1 - R) \rfloor, \tag{4.7}$$

where $R = K/N$ is the coding rate of the product code. Codes satisfying the equality in the above Singleton bound are referred to as diversity-wise MDS or block-fading MDS codes. From (4.7), we deduce that $R \leq 1/M$ if $L = M$ (full-diversity coding). For example, we get $R \leq 1/2$ with an edge coloring using $L = M = 2$ colors and $R \leq 1/4$ for $L = M = 4$ colors. The coding rate can exceed $1/M$ when $L < M$ in applications where full diversity is not mandatory. An example suited to distributed storage is an edge coloring with a palette of $M = 4$ colors, a diversity $L = 2$, and $R \leq 3/4$.

4.2.4 Rootcheck nodes and root symbols

In a way similar to root-LDPC codes and product codes built for block-fading channels [14][17], we introduce now the notion of root symbols and rootcheck nodes in product codes to be designed for distributive storage. A linear $[n, k]_q$ code with parity-check matrix H can fill $n - k$ erasures at positions where the columns of H are independent. These $n - k$ symbols correspond to $n - k$ separate edges in the non-compact graph and to a unique edge (supersymbol) in the compact graph. Therefore, for simplicity, we start by defining a root supersymbol in the compact graph where supernodes are equivalent to standard LDPC parity-check nodes.

Definition 4.5. Let $\mathcal{G}^c$ be a compact graph of a product code, let ϕ be a given edge coloring, and let $e \in E^c$ be a supersymbol. e is a *root supersymbol* with respect to $\phi(e)$ if it admits a neighbor vertex v, $v \in V_1^c$ or $v \in V_2^c$, such that all adjacent edges f in v satisfy $\phi(f) \neq \phi(e)$.

In Definition 4.5, if $v \in V_1^c$ then e is a root supersymbol thanks to the product code column to which it belongs, i.e. e can be solved in one iteration by its column component code when the color $\phi(e)$ is erased. Likewise, e is protected against erasures by its row component code if $v \in V_2^c$ in the previous definition. Finally, a root supersymbol may be

doubly protected by both its row and its column if both right and left neighbors $v_1 \in V_1^c$ and $v_2 \in V_2^c$ satisfy the condition of Definition 4.5.

Definition 4.6. Let $\mathcal{G}$ be a non-compact graph of a product code, let ϕ be a given edge coloring, and let $e \in E$ be a symbol. e is a *root symbol* with respect to $\phi(e)$ if it admits a neighbor vertex v such that:
$\phi(f) = \phi(e)$ for at most $n_2 - k_2 - 1$ adjacent edges f if $v \in V_1$, or
$\phi(f) = \phi(e)$ for at most $n_1 - k_1 - 1$ adjacent edges f if $v \in V_2$.

As mentioned in the paragraph before Definition 4.5, Definition 4.6 implies that the $n_i - k_i$ root symbols with the same color should belong to positions of independent columns in the parity-check matrix of the component code C_i. This constraint automatically disappears for MDS component codes since any set of $n_i - k_i$ columns of H_i has full rank.

4.2.5 The rootcheck order in product codes

Not all symbols of a product code are root symbols. Under iterative row-column decoding on channels with block erasures, some symbols may be solved in two decoding iterations or more. Some set of symbols may never be solved and are referred to as stopping sets [27][93][88]. Our study is restricted to erasing the symbols of one color out of M. Hence, the rest of this chapter is restricted to double diversity, $L = 2$. Absence of diversity is equivalent to $L = 1$. We establish now the root order ρ of a symbol. For root symbols satisfying definitions 4.5 and 4.6, the root order is $\rho = 1$. For symbols that can be solved after two decoding iterations, we set $\rho = 2$. The formal definition of the root order ρ can be written in the following recursive manner (for $\rho \geq 2$).

Definition 4.7. Let $\mathcal{G}^c$ be a compact graph of a product code, let $\phi \in \Phi(E^c)$ be an edge coloring, and let $e \in E^c$ be a super-symbol. e has *root order* $\rho(e) = \min(\rho_1, \rho_2)$ where:
1- Let $v_1 \in V_1^c$ be the column neighbor vertex of e. $\forall f$ adjacent to e in v_1 and $\phi(f) = \phi(e)$, we have $\rho(f) < \rho_1$.
2- Let $v_2 \in V_2^c$ be the row neighbor vertex of e. $\forall f$ adjacent to e in v_2 and $\phi(f) = \phi(e)$, we have $\rho(f) < \rho_2$.

The previous definition implies that $\rho(e) = 1$ if there exists no adjacent edge with the same color. Also, for an edge e that does not admit a finite $\rho(e)$, we set $\rho(e) = \infty$. When color $\phi(e)$ is erased, symbols belonging to the so-called stopping sets can never be solved (even after an infinite number of decoding iterations) and hence their root order is infinite. In the next section we review stopping sets as known in the literature and we study new stopping sets for product codes based on MDS components under iterative algebraic decoding. Definition 4.7 can be rephrased to make it suitable for the non-compact graph $\mathcal{G}$. We pursue this section to establish an upper bound of the largest finite root order valid for all edge colorings ϕ.

Theorem 4.1. *Let C_P be a product code $[n_1, k_1] \otimes [n_2, k_2]$ with a compact graph $\mathcal{G}^c = (V_1^c, V_2^c, E^c)$. $\forall \phi \in \Phi(E^c)$ and $\forall e \in E^c$ we have:*
Case 1: $\nexists f \in E^c$ such that $\phi(f) = \phi(e)$ and $\rho(f) = \infty$, then

$$1 \leq \rho(e) \leq \left\lceil \frac{N^c}{2M} \right\rceil = \rho_u.$$

Define the minimum number of good edges,

$$\eta_{min}(\phi) = \min_{i=1\ldots M} |\{f \in E^c : \phi(f) = i, \rho(f) = 1\}|.$$

Then, in Case 1,

$$2\rho(e) + \eta_{min}(\phi) - 3 \leq \left\lceil \frac{N^c}{M} \right\rceil. \tag{4.8}$$

Case 2: $\exists f \in E^c$ such that $\phi(f) = \phi(e)$ and $\rho(f) = \infty$, then

$$\rho(e) = \infty \quad or \quad 1 \leq \rho(e) \leq \left\lceil \frac{N^c}{M} \right\rceil - 4,$$

where $N^c = |E^c|$ is given by (4.2).

Proof. Case 1 corresponds to a product code with diversity $L = 2$, for a given color $\phi(e)$, which is capable of solving all symbols when that color is erased. The graph has no infinite root order symbols. ρ is recursively built by starting from $\rho = 1$ following two paths in the graph until reaching a common edge e that has two neighboring vertices with edges of order $\rho(e) - 1$. There are up to $\lceil N^c/M \rceil$ edges, including e, having color equal to $\phi(e)$. The largest $\rho(e)$ is attained in the middle of the longest path of length $\lceil N^c/M \rceil$, hence $2\rho(e) - 1 \leq \lceil N^c/M \rceil$ which is translated into the stated result for Case 1. An illustrated instance is given for the reader in Example 4.1. Back to the path of length $2\rho(e) - 1$ ending with edges of order 1 on both sides, if the population of order 1 edges is η_1 for the color $\phi(e)$, then the path can only use a maximum of $\lceil N^c/M \rceil - (\eta_1 - 2)$ edges. We get the inequality $2\rho(e) - 1 \leq \lceil N^c/M \rceil - (\eta_1 - 2)$. By plugging $\eta_{min}(\phi)$ instead of η_1, this inequality becomes independent from the particular color. The stated inequality in (4.8) is obtained after grouping $\rho(e)$ and $\eta_{min}(\phi)$ on the left side.
Case 2 corresponds to bad edge coloring where the product code does not have double diversity, i.e. stopping sets do exist for the color $\phi(e)$. The order of e may be infinite if e is involved in a stopping set with another edge f having the same color. Otherwise, consider the smallest stopping set of size four symbols (the smallest cycle in $\mathcal{G}^c$ with edges of color $\phi(e)$), then there remains $\lceil N^c/M \rceil - 4$ edges of color $\phi(e)$. A path of length $\lceil N^c/M \rceil - 4$ starting with $\rho = 1$ and ending at $\rho = \infty$ may exist. The largest finite order in this path before reaching the stopping set is $\rho = \lceil N^c/M \rceil - 4$. □

Corollary 4.1. *Let C_P be a product code $[n_1, k_1] \otimes [n_2, k_2]$ with a compact graph $\mathcal{G}^c$. Let $\phi \in \Phi(E^c)$ be an edge coloring. We define*

$$\rho_{max}(\phi) = \max_{e \in E^c} \rho(e). \tag{4.9}$$

C_P attains double diversity under iterative row-column decoding if and only if $\rho_{max}(\phi) < \infty$. In this case, we say that ϕ is a double-diversity coloring and $\forall e \in E^c$, e can be solved after at most ρ_{max} decoding iterations where $\rho_{max}(\phi) \leq \rho_u$.

For colorings in $\Phi(E)$, we extend the same definition as in Corollary 4.1 and we say that $\phi \in \Phi(E)$ is double-diversity if all edges have a finite rootcheck order. The parameter ρ_{max} is important in practical applications to bound from above the amount of conveyed information within a network (whether it is a local-area or a wide-area network). In fact, in coding for distributed storage, the locality of a product code per decoding iteration is $\max(n_1, n_2)$ in $\mathcal{G}$ under algebraic decoding of its row and column components. Here, the locality is the number of symbols to be accessed in order to repair an erased symbol, refer to Definition 2.1. Locality is $\max(k_1, k_2)$ for MDS components under ML decoding of the product code components. Finally, for a product code, the information transfer per symbol is bounded from above by

$$\rho_{max}(\phi) \times \max(n_1, n_2). \tag{4.10}$$

The exact transfer cost to fill all erasures with iterative decoding can be determined by multiplying each order ρ with the corresponding edge population size. This exact cost may vary in a wide range from one coloring to another. The DECA algorithm presented in Section 4.4 dramatically reduces ρ_{max} by enlarging the edge population with root order 1. The interdependence between ρ and the population of order 1 was revealed in inequality (4.8). This inequality is useful in intermediate cases where $\rho_{max} = 1$ is not attained, i.e. outside the case where all edges have order 1. The influence of the component decoding method on the performance of a product code via its stopping sets is discussed in Section 4.3.

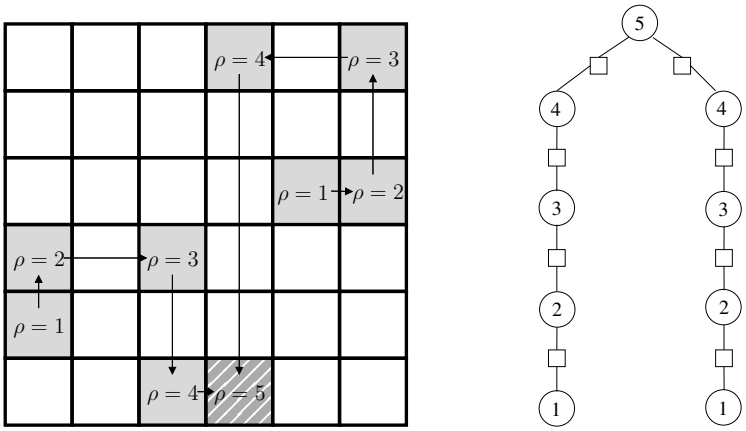


Figure 4.3: Compact matrix (left) and path in compact graph (right) for a product code $[12, 10]^{\otimes 2}$ showing a maximal root order of 5.

Example 4.1. Consider a $[12, 10]^{\otimes 2}$ product code and a coloring ϕ with $M = 4$ colors. The compact graph has $|E^c| = 6 \times 6$ edges. Instead of drawing $\mathcal{G}^c$, we draw the 6×6 compact matrix representation of the product code in Fig. 4.3. Supersymbols corresponding to a color $\phi(e) = 1$ are shaded. Fig. 4.3 also shows a path in $\mathcal{G}^c$ such that a maximal order $\rho_{max} = \rho_u = 5$ is attained for $\phi(e) = 1$. If ϕ has double diversity then ρ_{max} will not exceed $\rho_u = 5$ for all colors $\phi(e) \in \{1, 2, \ldots, M\}$. Note that the parameters of this product code are such that $N^c/M - 4$ is also equal to 5 for a ϕ with a diversity defect.

Example 4.2. Consider a $[14, 12] \otimes [16, 14]$ product code and a coloring ϕ with $M = 4$ colors. The compact graph has $|E^c| = 7 \times 8$ edges. The compact matrix and a path attaining $\rho = 10$ are illustrated in Fig. 4.4. ϕ is chosen such that the first color has a cycle involving four supersymbols. Starting from the root supersymbol ($\rho = 1$) it is possible to create a path in the graph such that $\rho = 10$ is reached. Note that a double-diversity ϕ cannot exceed a root order $\rho_u = 7$.

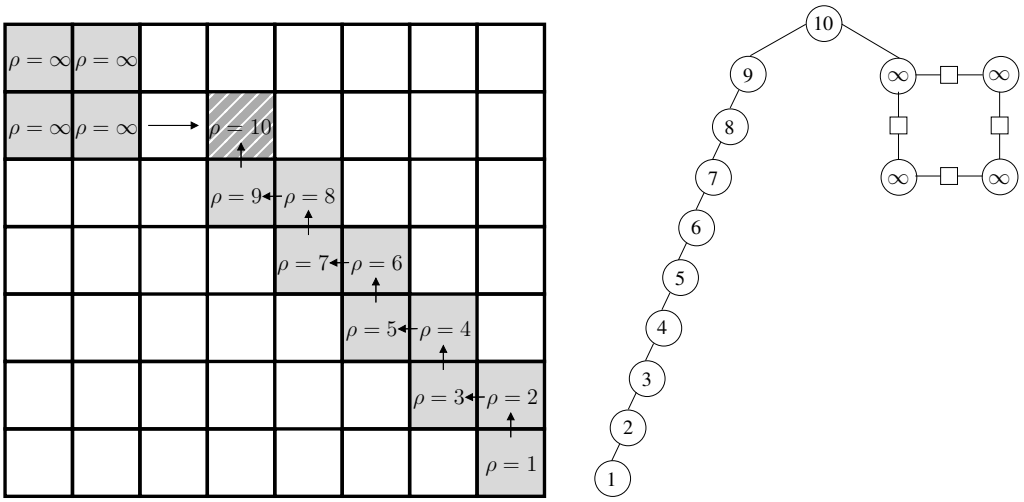


Figure 4.4: Compact matrix (left) and path in compact graph (right) for a product code $[14, 12] \otimes [16, 14]$ showing a maximal finite root order of 10.

The ideal situation is to construct a product code and its edge coloring in order to obtain $\rho(e) = 1$ for all edges. We investigate now the conditions on the product code rate and its components rates in this ideal situation. The analysis based on ρ_u reveals the existence of a trade-off between minimizing the number of decoding iterations and the valid range of both coding rates for the product code components.

Firstly, let us look at the upper bound ρ_u from Theorem 4.1. Without loss of generality, assume that $n_i - k_i$ divides n_i. Then, we have

$$R_i = 1 - \frac{n_i - k_i}{n_i} = 1 - \frac{1}{|V_i^c|}. \tag{4.11}$$

The total coding rate becomes

$$R = R_1 R_2 = \left(1 - \frac{1}{|V_1^c|}\right) \cdot \left(1 - \frac{1}{|V_2^c|}\right). \tag{4.12}$$

Using $N^c = |V_1^c| \cdot |V_2^c|$, we get

$$R_1 R_2 = R_1 + R_2 - 1 + \frac{1}{N^c}. \tag{4.13}$$

Finally, from (4.13) and Theorem 4.1, the upper bound of the root order for double-diversity edge coloring of the compact graph can be expressed as

$$\rho_u = \left\lceil \frac{N^c}{2M} \right\rceil = \left\lceil \frac{1}{2M \times (1 + R_1 R_2 - R_1 - R_2)} \right\rceil. \tag{4.14}$$

Fix the product code rate R, force the upper bound to $\rho_u = 1$, and take $M = 4$ colors. Then the denominator in (4.14) should be less than 1 or equivalently $-8R_1^2 + (7+8R)R_1 - 8R > 0$. This second-degree polynomial in R_1 is non-negative if and only if

$$R < \frac{9}{8} - \frac{1}{\sqrt{2}} \approx 0.4178, \tag{4.15}$$

and

$$-\sqrt{64R^2 - 144R + 49} < 16R_1 - 8R - 7 < +\sqrt{64R^2 - 144R + 49}. \tag{4.16}$$

As a result, with a palette of four colors, (4.15) tells us that $\rho(e) = 1$ for all edges is feasible for a product code with a rate less than 0.4178. It is obvious that (4.15) is a very constraining condition because ρ_u is an upper bound of $\rho_{max}(\phi)$ for all $\phi \in \Phi(E^c)$. It is worth noting that R_1 and R_2 vary in a smaller range when R approaches $\frac{9}{8} - \frac{1}{\sqrt{2}}$, which corresponds to a product code with balanced components.

In Section 4.4.1, we will show unbalanced product codes where a sufficient condition on the component rates imposes order 1 to all edges. The sufficient condition, not based on ρ_u, is given by Lemma 4.5. But before introducing an efficient edge coloring algorithm in Section 4.4, we analyze stopping sets in product codes with MDS components in the next section, we describe the relationship between stopping sets and the product code graph representation, and finally we enumerate obvious and non-obvious stopping sets. Stopping sets enumeration is useful to determine the performance of a product code with and without edge coloring.

4.3 Stopping sets for MDS components

The purpose of this section is to prepare the way for determining the performance of iterative decoding of non-binary product codes. The analysis of stopping sets in a product code will yield a tight upper bound of its iterative decoding performance over a channel with independent erasures. The same analysis will be useful to accurately estimate the performance under edge coloring in presence of block and multiple erasure channels.

4.3.1 Decoding erasures

Definition 4.8. An erasure pattern is said to be ML-correctable if the ML decoder is capable of solving all its erased symbols.

For an erasure pattern which is not correctable under ML or iterative decoding, the decoding process may fill none or some of the erasures and then stay stuck on the remaining ones. Before describing the stopping sets of a product code, let us recall some fundamental results regarding the decoding of its row and column component codes. The ML erasure-filling capability of a linear code satisfies the following property.

Proposition 4.1. *Let $C[n,k,d]_q$ be a linear code with $q \geq 2$. Assume that C is not MDS and the n symbols of a codeword are transmitted on an erasure channel. Then, there exists an erasure pattern of weight greater than $d-1$ that is ML-correctable.*

Proof. Let H be an $(n-k)\times n$ parity-check matrix of C with rank $n-k > d-1$. For any integer w in the range $[d, n-k]$, there exists a set of w linearly independent columns in H. Choose an erasure pattern of weight w with erasures located at the positions of the w independent columns. Then, the ML decoder is capable of solving all these erasures by simple Gaussian reduction of H. □

For MDS codes, based on a proof similar to the proof of Proposition 4.1, we state a well-known result in the following corollary.

Corollary 4.2. *Let $C[n,k,d]_q$ be an MDS code. All erasure patterns of weight greater than $d-1$ are not ML-correctable.*

We conclude from the previous corollary that an algebraic decoder for an MDS code attains the word-error performance of its ML decoder. What about symbol-error performance? Indeed, for general binary and non-binary codes, the ML decoder may outperform an algebraic decoder since it is capable of filling some of the erasures when dealing with a pattern which is not ML-correctable. In the MDS case, the answer comes from the absence of spectral holes for any MDS code beyond its minimum distance. This basic result is proven via standard tools from algebraic coding theory [65][10]:

Proposition 4.2. *Let $C[n,k,d]_q$ be a non-binary MDS code ($q > n > 2$). For any w satisfying $d \leq w \leq n$ and any support $\mathcal{X} = \{i_1, i_2, \ldots, i_w\}$, where $1 \leq i_j \leq n$, there exists a codeword in C of weight w having $\mathcal{X}$ as its own support.*

Proof. By assumption we have $w > r = n-k$. Let H be a parity-check matrix of C with rank $r = n-k$. Recall that the MDS property makes full-rank any set of $n-k$ columns of H [65]. w is written as $w = r+\ell$, where $\ell = 1 \ldots k$. The w positions of $\mathcal{X}$ are anywhere inside the range $[1,n]$, but for simplicity let us denote $h_1 \ldots h_r$ the r columns of H in the first r positions. The last ℓ columns are denoted $\zeta_1 \ldots \zeta_\ell$. For any $j = 1 \ldots \ell$, we have

$$\zeta_j = \sum_{i=1}^{r} a_{i,j} h_i,$$

where $a_{i,j} \in \mathbb{F}_q \setminus \{0\}$ otherwise it contradicts $d = n - k + 1$. Now, select $\alpha_1 \dots \alpha_\ell$ from $\mathbb{F}_q \setminus \{0\}$ such that: α_1 is arbitrary, α_2 is chosen outside the set $\{-\alpha_1 a_{i,1}/a_{i,2}\}_{i=1}^{r}$, then α_3 is chosen outside the set $\{(-\alpha_1 a_{i,1} - \alpha_2 a_{i,2})/a_{i,3}\}_{i=1}^{r}$, and so on, up to α_ℓ which is chosen outside the set $\{-\sum_{u=1}^{\ell-1} \alpha_u a_{i,u}/a_{i,\ell}\}_{i=1}^{r}$. Here, the notation a/b in $\mathbb{F}_q \setminus \{0\}$ is equivalent to the standard algebraic notation ab^{-1}. The equality

$$\sum_{j=1}^{\ell} \alpha_j \zeta_j = \sum_{i=1}^{r} \sum_{j=1}^{\ell} \alpha_j a_{i,j} h_i$$

produces a codeword of Hamming weight w. Hence, there exists a codeword of weight w with non-zero symbols in all positions given by $\mathcal{X}$. □

Now, at the symbol level for an MDS code and an erasure pattern which is not ML-correctable ($w > d - 1$), we conclude from Proposition 4.2 that the ML decoder cannot solve any of the w erasures because they are covered by a codeword. Consequently, an algebraic decoder for an MDS code also attains the symbol-error performance of the ML decoder. This behavior will have a direct consequence on the iterative decoding of a product code with MDS components: stopping sets are identical when dealing with algebraic and ML-per-component decoders.

A general description of a stopping set was given by Definition 4.1. The exact definition of a stopping set depends on the iterative decoding type. For product codes, four decoding methods are known:

- Type I: ML decoder. This is a non-iterative decoder. It is based on a Gaussian reduction of the parity-check matrix of the product code.
- Type II: Iterative algebraic decoder. At odd decoding iterations, component codes C_1 on each column are decoded via an algebraic decoder (bounded-distance) that fills up to $d - 1$ erasures. Similarly, at even decoding iterations, component codes C_2 on each row are decoded via an algebraic decoder.
- Type III: Iterative ML-per-component decoder. This decoder was considered by Rosnes in [88] for binary product codes. At odd decoding iterations, column codes C_1 are decoded via an optimal decoder (ML for C_1). At even decoding iterations, row codes C_2 are decoded via a similar optimal decoder (ML for C_2).
- Type IV: Iterative belief-propagation decoder based on the Tanner graph of $C_{\mathcal{P}}$, as studied by Schwartz et al. for general linear block codes [93] and by Di et al. for low-density parity-check codes [27].

The three iterative decoders listed above give rise to three different kinds of stopping sets. As previously indicated, from Corollary 4.2 and Propositions 4.2, we concluded that type-II and type-III stopping sets are identical if component codes are MDS.

4.3.2 Stopping set definition

Let C be a q-ary linear code of length n, i.e. C is a sub-space of dimension k of $\mathbb{F}_q^n$. The support of C, denoted by $\mathcal{X}(C)$, is the set of ℓ distinct positions $\{i_1, i_2, \ldots, i_\ell\} = \{i_j\}_{j=1}^{\ell}$, $1 \leq i_j \leq n$, such that, for all j, there exists a codeword $c = (c_1 \ldots c_n) \in C$ with $c_{i_j} \neq 0$. This notion of support $\mathcal{X}$ is applied to rows and columns in a product code.

Now, we define a rectangular support which is useful to represent a stopping set in a bi-dimensional product code. Let $\mathcal{S} \subseteq \{1, \ldots, n_1\} \times \{1, \ldots, n_2\}$ be a set of symbol positions in the product code. The set of row positions associated to $\mathcal{S}$ is $\mathcal{R}_1(\mathcal{S}) = \{i_1, \ldots, i_{\ell_1}\}$ where $|\mathcal{R}_1(\mathcal{S})| = \ell_1$ and for all $i \in \mathcal{R}_1(\mathcal{S})$ there exists $(i, \ell) \in \mathcal{S}$. The set of column positions associated to $\mathcal{S}$ is $\mathcal{R}_2(\mathcal{S}) = \{j_1, \ldots, j_{\ell_2}\}$ where $|\mathcal{R}_2(\mathcal{S})| = \ell_2$ and for all $j \in \mathcal{R}_2(\mathcal{S})$ there exists $(\ell, j) \in \mathcal{S}$. The rectangular support of $\mathcal{S}$ is

$$\mathcal{R}(\mathcal{S}) = \mathcal{R}_1(\mathcal{S}) \times \mathcal{R}_2(\mathcal{S}), \tag{4.17}$$

i.e. the smallest $\ell_1 \times \ell_2$ rectangle including all columns and all rows of $\mathcal{S}$.

Definition 4.9. Consider a product code $C_P = C_1 \otimes C_2$. Let $\mathcal{S} \subseteq \{1, \ldots, n_1\} \times \{1, \ldots, n_2\}$ with $|\mathcal{R}_1(\mathcal{S})| = \ell_1$ and $|\mathcal{R}_2(\mathcal{S})| = \ell_2$. Consider the ℓ_1 rows of $\mathcal{S}$ given by $\mathcal{S}_r^{(i)} = \{j : (i, j) \in \mathcal{S}\}$ and the ℓ_2 columns of $\mathcal{S}$ given by $\mathcal{S}_c^{(j)} = \{i : (i, j) \in \mathcal{S}\}$. The set $\mathcal{S}$ is a stopping set of type III for C_P if there exist linear subcodes $C_c^{(j)} \subseteq C_1$ and $C_r^{(i)} \subseteq C_2$ such that $\mathcal{X}(C_c^{(j)}) = \mathcal{S}_c^{(j)}$ and $\mathcal{X}(C_r^{(i)}) = \mathcal{S}_r^{(i)}$ for all $i \in \mathcal{R}_1(\mathcal{S})$ and for all $j \in \mathcal{R}_2(\mathcal{S})$.

The cardinality $|\mathcal{S}|$ is called the size of the stopping set and will also be referred to in the sequel as the weight of $\mathcal{S}$. Recall that type II and type III stopping sets are identical when both C_1 and C_2 are MDS. Stopping sets of type III were studied for binary product codes by Rosnes [88]. His analysis is based on the generalized Hamming distance [114][44] because sub-codes involved in Definition 4.9 may have a dimension greater than 1. In the non-binary MDS case, according to Proposition 4.2, all these sub-codes have dimension 1, i.e. they are generated by a single non-zero codeword. Consequently, the generalized Hamming distance is not relevant when using MDS components. In such a case, the analysis of type II stopping sets is mainly combinatorial and does not require algebraic tools.

Stopping sets for decoder types II-IV can be characterized by four main properties summarized as follows.

- Obvious or not obvious sets, also known as rank-1 sets. A stopping set $\mathcal{S}$ is obvious if $\mathcal{S} = \mathcal{R}(\mathcal{S})$.
- Primitive or non-primitive stopping sets. A stopping set is primitive if it cannot be partitioned into two or more smaller stopping sets. Notice that all stopping sets, whether they are primitive or not, are involved in the code performance.
- Codeword or non-codeword. A stopping set $\mathcal{S}$ is said to be a codeword stopping set if there exists a codeword c in C_P such that $\mathcal{X}(c) = \mathcal{S}$.

- ML-correctable or non-ML-correctable. A stopping set $\mathcal{S}$ cannot be corrected via ML decoding if it includes the support of a non-zero codeword.

In the remaining material of this chapter, we restrict our study to type II stopping sets.

Example 4.3. Consider a $[n_1, n_1 - 2, 3]_q \otimes [n_2, n_2 - 2, 3]_q$ product code. A stopping set $\mathcal{S}$ of size $w = 9$ is shown as a weight-9 matrix of size $n_1 \times n_2$, where 1 corresponds to an erased position:

$$\mathcal{S} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{4.18}$$

We took $n_1 = n_2 = 7$ for illustration. The rectangular support is shown in a compact representation as a matrix of size $\ell_1 \times \ell_2 = 3 \times 3$,

$$\mathcal{R}(\mathcal{S}) = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}. \tag{4.19}$$

The stopping set in (4.18) is obvious, it has the same size as its rectangular support. It corresponds to a matrix of rank 1. Each row and each column of $\mathcal{S}$ has weight 3. Iterative row-column decoding based on component algebraic decoders fails in decoding rows and columns since the number of erasures exceeds the erasure-filling capacity of the MDS components. This stopping set is not ML-correctable because it is a product-code codeword. In the sequel, all stopping sets (type II) shall be represented in this compact manner by a smaller rectangle of size $\ell_1 \times \ell_2$.

Example 4.4. For the same $[n_1, n_1 - 2, 3]_q \otimes [n_2, n_2 - 2, 3]_q$ product code used in the previous example, the following stopping sets of size 12 are not obvious.

$$\mathcal{S}_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{4.20}$$

$$\mathcal{S}_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{4.21}$$

In compact form, their rectangular support is

$$\mathcal{R}(\mathcal{S}_1) = \mathcal{R}(\mathcal{S}_2) = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix}. \tag{4.22}$$

These stopping sets have size 12 and a 4×4 rectangular support. For $w = 12$, it is also possible to build an obvious stopping set in a 3×4 rectangle or a 4×3 rectangle full of 1. $\mathcal{S}_1$ is ML-correctable since it does not cover a product code codeword. $\mathcal{S}_2$ covers a codeword hence it is not ML-correctable.

4.3.3 Stopping sets and subgraphs of product codes

A stopping set as defined by Definition (4.9) corresponds to erased edges in the non-compact graph $\mathcal{G}$ introduced in Section 4.2.1. Indeed, consider the size-9 stopping set given by (4.18) or (4.19). The nine symbol positions involve nine edges in $\mathcal{G}$, three row check nodes, and three column check nodes. Each of these six check nodes has three erased symbols making the $[12, 10, 3]$ decoder fail. This stopping set is equivalent to a subgraph of 9 edges in $\mathcal{G}$ as shown in Figure 4.5. The subgraph in Figure 4.5 has three length-4 cycles and two length-6 cycles. The small cycles of length-4 are associated to an erasure pattern with a 2×2 rectangular support which is not a stopping set ($d_1 = d_2 = 3$). Similarly, length-6 cycles are not stopping sets and are associated to erasure patterns with a 2×3 rectangular support. We will see in the next section that the minimum stopping set size is $d_1 d_2 = 9$, i.e. it is equal to the minimum Hamming distance of the product code.

A subgraph of $\mathcal{G}^c$ can be embedded into $\mathcal{G}$ by splitting each super-edge into $(n_1 - k_1) \times (n_2 - k1)$ edges. The converse is not always true. The subgraph with nine edges in Figure 4.5 cannot be compressed into a subgraph of $\mathcal{G}^c$. For the $[12, 10, 3]^{\otimes 2}$ product code, a supersymbol in $\mathcal{G}^c$ contains four edges. Hence, a necessary condition for a stopping set in $\mathcal{G}$ to become a valid stopping set in $\mathcal{G}^c$ is to erase edges in groups of 4. Knowing that type II and type III stopping sets are identical when row and column codes C_1 and C_2 are MDS, Definition (4.9) leads to the following corollaries.

Corollary 4.3. *Let $C_P = C_1 \otimes C_2$ be a product code with MDS components C_1 and C_2 having minimum Hamming distance d_1 and d_2 respectively. Assume that symbols*

(edges) of $\mathcal{G} = (V_1, V_2, E)$ are sent over an erasure channel. A stopping set for the iterative decoder is a subgraph of $\mathcal{G}$ such that all column vertices in V_1 have a degree greater than or equal to d_1 and all row vertices in V_2 have a degree greater than or equal to d_2.

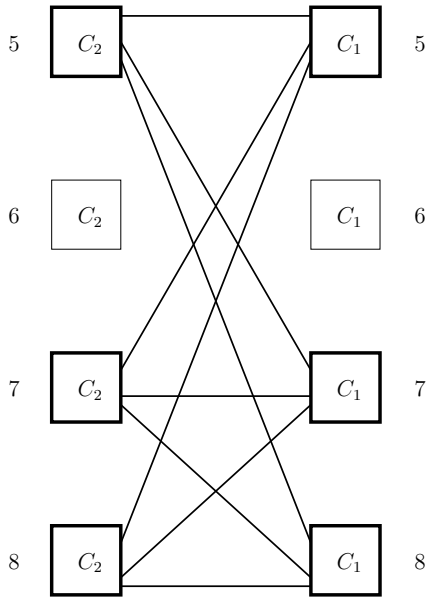


Figure 4.5: A sub-graph of $\mathcal{G}$ representing the size-9 obvious stopping set. The graph $\mathcal{G}$ has $|E| = 144$ edges, $|V_2| = 12$ left (row) check nodes, and $|V_1| = 12$ left (column) check nodes. Only the stopping set edges are drawn.

Corollary 4.4. *Let $C_P = C_1 \otimes C_2$ be a product code with MDS components C_1 and C_2 having minimum Hamming distance d_1 and d_2 respectively. Assume that supersymbols (super-edges) of $\mathcal{G}^c = (V_1^c, V_2^c, E^c)$ are sent over an erasure channel. A stopping set for the iterative decoder is a subgraph of $\mathcal{G}^c$ such that all column vertices in V_1^c have a degree greater than or equal to 2 and all row vertices in V_2 have a degree greater than or equal to 2.*

The above corollaries suppose a symbol (or a supersymbol) channel with independent erasures. When $\mathcal{G}$ is endowed with an edge coloring ϕ, we get the same constraint on the validity of a subgraph embedding from $\mathcal{G}^c$ into $\mathcal{G}$. We know from Section 4.2.1 that $\Phi(E^c \to E)$ is a subset of $\Phi(E)$, i.e. some edge colorings of $\mathcal{G}$ are not edge colorings of $\mathcal{G}^c$. Consequently, on a block-erasure channel, if all super-edges of the same color are erased, stopping sets in $\mathcal{G}^c$ are a subset of those in $\mathcal{G}$. The non-compact graph $\mathcal{G}$ has a larger ensemble of stopping sets, with or without edge coloring. As an example, for the $[12, 10, 3]^{\otimes 2}$ product code, the smallest stopping set in $\mathcal{G}^c$ has size 2×2 when four super-edges are erased which yields a stopping set of size 16 in $\mathcal{G}$.

Example 4.5. Consider the $[9,6,4]_q^{\otimes 2}$ product code where $d_1 = d_2 = 4$ and $q > 9$. Assume that our palette has $M = 3$ colors. The non-compact graph admits an ensemble of $|\Phi(E)| = 449018638290329886295066893074864640$ edge colorings! The compact graph has $|\Phi(E^c)| = 1680$ only. In $\mathcal{G}^c$, each color is used $N^c/M = 3$ times. For a channel erasing all symbols of the same color, the compact graph has no stopping sets (the 2×2 rectangular support cannot be filled by a single color). A compact matrix representation of $\mathcal{G}^c$ attaining double diversity with all symbols of order 1 is given by the trivial matrix

$$\begin{bmatrix} R & G & B \\ B & R & G \\ G & B & R \end{bmatrix}, \tag{4.23}$$

where the color $\phi(e) = 1$ is replaced by the letter 'R', $\phi(e) = 2$ is replaced by the letter 'G', and $\phi(e) = 3$ is replaced by the letter 'B'. The non-compact graph has 9×9 edges, each color is used 27 times. Double diversity is lost in $\mathcal{G}$ if one of the 4×4, 4×5, or 5×5 obvious stopping sets is covered by a unique color. Clearly, $\mathcal{G}^c$ makes the design much easier. This double-diversity product code has a relatively low coding rate. More challenging product code designs are given in Section 4.4 with higher rates up to the one imposed by the block-fading/block-erasure Singleton bound.

4.3.4 Enumeration of stopping sets

For a fixed non-zero integer w, the number of stopping sets of size w, denoted as τ_w, falls in two different cases. Firstly, $\tau_w = 0$ if w is small with respect to the minimum Hamming distance of the product code. Also, $\tau_w = 0$ for special erasure patterns obtained by adding a small neighborhood to a smaller obvious set. Secondly, for both obvious and non-obvious stopping sets, τ_w is non-zero and the weight w may correspond to many rectangular supports of different height and width. The code performance over erasure channels is dominated by not-so-large stopping sets. Non-empty stopping sets of the second case satisfy the general property stated in the following lemma.

Lemma 4.1. *Given a weight $w \leq (d_1+1)(d_2+1)$ and assuming $\tau_w > 0$, then $\exists \mathcal{S}^0$ such that $\forall \mathcal{S}$ with $|\mathcal{S}| = w$, we have $\|\mathcal{R}(\mathcal{S})\| \leq \|\mathcal{R}(\mathcal{S}^0)\| = (\ell_1^0, \ell_2^0)$, where*

$$\ell_1^0 \leq d_1 + 1 + \left\lfloor \frac{d_1+1}{d_2} \right\rfloor, \tag{4.24}$$

$$\ell_2^0 \leq d_2 + 1 + \left\lfloor \frac{d_2+1}{d_1} \right\rfloor. \tag{4.25}$$

Proof. Let w be equal to $(d_1+1)(d_2+1)$. In order to establish an upper bound of the height ℓ_1, we build the highest possible rectangular support for this weight w. Assume the rectangle is $\ell_1^0 \times \ell_2$, each of its rows should have at least d_2 erasures to make the type II decoder fail. Then $d_2 \ell_1^0 \leq (d_1+1)(d_2+1)$, which becomes the upper bound given by (4.24). Now, if w is less than $(d_1+1)(d_2+1)$, the rectangular support of the stopping set can only shrink in size. The upper bound of the width in (4.25) is proven in a similar way. □

The above lemma states the existence of a maximal rectangular support for a given stopping set size. The example given below cites stopping sets with a unique-size rectangular support and stopping sets with multiple-size rectangular supports.

Example 4.6. Consider a $C_1 \otimes C_2$ product code where C_1 and C_2 are both MDS with minimum Hamming distance 3. The stopping set given by (4.19) cannot have a large rectangular support. In general, all stopping sets of size $d_1 d_2$ have a rectangular support of fixed dimensions $d_1 \times d_2$. Now, let $w = 12$. As indicated in Example 4.4, stopping sets of size 12 may be included in rectangular supports of dimensions 3×4, 4×3, and 4×4. For $w = 12$, it is impossible to build a 4×5 rectangular support (reductio ad absurdum) making $\ell_1^0 = 4$ and $\ell_2^0 = 4$. A similar proof by contradiction yields $\ell_1^0 = 5$ and $\ell_2^0 = 5$ for $w = 15$.

The next lemma gives an obvious upper bound of the size of $\mathcal{R}(\mathcal{S})$ by stating a simple limit on the number of zeros (non-erased positions) inside $\mathcal{R}(\mathcal{S})$.

Lemma 4.2. *Let $\mathcal{R}(\mathcal{S})$ be the $\ell_1 \times \ell_2$ rectangular support of a stopping set $\mathcal{S}$ of size w. Let $\beta = \ell_1 \ell_2 - w$ be the number of zero positions, or equivalently β is the size of the set $\mathcal{R}(\mathcal{S}) \setminus \mathcal{S}$. Then*

$$\beta \leq \min((\ell_1 - d_1)\ell_2, \ell_1(\ell_2 - d_2)). \tag{4.26}$$

Before stating and proving Theorem 4.2, we announce two results in Lemma 4.3 and Lemma 4.4 on bipartite graphs enumeration. We saw in the previous section that stopping sets are sub-graphs of $\mathcal{G}$ and $\mathcal{G}^c$, see Corollary 4.3 and Corollary 4.4. In other words, the enumeration of stopping sets represented as matrices of a given distribution of row weight and column weight is equivalent to enumerating bipartite graphs where left vertices stand for rows and right vertices stand for columns. An edge should be drawn between a left vertex and a right vertex according to some rule, e.g. the rule used in the previous section draws an edge in the bipartite graph for each 1 in the stopping set matrix. Stopping sets enumeration in the next theorem is based on β, the number of zeros or the number of non-erased positions. Hence, we shall use the opposite rule. A stopping set of weight w and having a $\ell_1 \times \ell_2$ rectangular support shall be represented by a bipartite graph with ℓ_1 left vertices, ℓ_2 right vertices, and a total of $\beta = \ell_1 \ell_2 - w$ edges. Notice that these bipartite graphs have no length-2 cycles because parallel edges are forbidden.

For finite ℓ_1 and ℓ_2, given the left degree distribution and the right degree distribution, there exists no exact formula for counting bipartite graphs. The best recent results are asymptotic in the graph size for sparse and dense matrices [20] [24] and cannot be applied in our enumeration. The following two lemmas solve two cases encountered in Theorem 4.2 for $w = d(d+2)$ and $w = (d+1)(d+1)$ both inside a $(d+2) \times (d+2)$ rectangular support. The definition of special partitions is required before introducing the two lemmas.

Definition 4.10. Let $\ell \geq 2$ be an integer. A *special partition* of length j of ℓ is a partition defined by a tuple $(\ell_1, \ell_2, \ldots, \ell_j)$ such that its integer components satisfy:

- $\ell_1 \leq \ell_2 \leq \ldots \leq \ell_j$.
- $\sum_{i=1}^{j} \ell_i = \ell$.
- $\ell_i \geq 2$, $\forall j$.
- $1 \leq j \leq \ell/2$.

A special partition shall be denoted by $((\ell_1, \ldots, \ell_j))$.

Definition 4.11. The *group number* of a special partition, denoted by $\kappa = \kappa(\ell_1, \ell_2, \ldots, \ell_j)$, is the number of different integers ℓ_j, for $j = 1 \ldots \ell/2$. In other words, following set theory, the set including the j integers ℓ_i's is $\{\ell_{i_1}, \ell_{i_2}, \ldots, \ell_{i_\kappa}\}$. The group number divides the partition of ℓ into κ groups where the m^{th} group includes ℓ_{i_m} repeated g_m times, and $\sum_{m=1}^{\kappa} g_m = j$.

Lemma 4.3. *Consider bipartite graphs defined as follows: ℓ left vertices, ℓ right vertices, all vertices have degree 2, and no length-2 cycles are allowed. For $\ell \geq 2$, the total number x_ℓ of such bipartite graphs is given by the expression*

$$x_\ell = \sum_{((\ell_1,\ldots,\ell_j))} \frac{1}{\prod_{m=1}^{\kappa(\ell_1,\ldots,\ell_j)} g_m!} \prod_{k=1}^{j} \frac{\prod_{u=0}^{\ell_k-1} (\ell - \sum_{i=1}^{k-1} \ell_i - u)^2}{2\ell_k} \tag{4.27}$$

where $\sum_{((\ell_1,\ldots,\ell_j))}$ is a summation over all special partitions of the integer ℓ, $\kappa(\ell_1, \ldots, \ell_j)$ is the group number of the special partition $((\ell_1, \ldots, \ell_j))$, and g_m is the size of the m^{th} group.

Proof. Firstly, let us find the number of Hamiltonian bipartite graphs having ℓ_k left vertices, ℓ_k right vertices, all vertices of degree 2, and no length-2 cycles allowed. There are $(\ell_k!)^2$ ways to choose the order of all left and right vertices. If the Hamiltonian cycle is represented by a sequence of $2\ell_k$ integers corresponding to the $2\ell_k$ vertices of the bipartite graph, then there are $2\ell_k$ ways to shift the Hamiltonian cycle without changing the graph. Hence, the number of Hamiltonian bipartite graphs of degree 2 is

$$\frac{(\ell_k!)^2}{2\ell_k}. \tag{4.28}$$

Secondly, given the half-size ℓ of the bipartite graph stated in this lemma, all special partitions of ℓ are considered. For a fixed special partition $((\ell_1, \ell_2, \ldots, \ell_j))$ the bipartite graph is decomposed into j Hamiltonian graphs each of length ℓ_k, $k = 1 \ldots j$. The number of choices for selecting the vertices of the j Hamiltonian graphs is

$$\prod_{k=1}^{j} \binom{\ell - \sum_{i=1}^{k-1} \ell_i}{\ell_k}^2. \tag{4.29}$$

The above number should be multiplied by the number of Hamiltonian graphs for each selection of vertices to get

$$\prod_{k=1}^{j} \binom{\ell - \sum_{i=1}^{k-1} \ell_i}{\ell_k}^2 \frac{(\ell_k!)^2}{2\ell_k} \tag{4.30}$$

But for a given special partition, each group of size g_m is creating $g_m!$ identical bipartite graphs. Hence, the final result for a fixed partition becomes

$$\frac{1}{\prod_{m=1}^{\kappa(\ell_1,\ldots,\ell_j)} g_m!} \prod_{k=1}^{j} \binom{\ell - \sum_{i=1}^{k-1} \ell_i}{\ell_k}^2 \frac{(\ell_k!)^2}{2\ell_k} \tag{4.31}$$

Then, x_ℓ is obtained by summing (4.31) over all special partitions of the integer ℓ to yield

$$x_\ell = \sum_{((\ell_1,\ldots,\ell_j))} \frac{1}{\prod_{m=1}^{\kappa(\ell_1,\ldots,\ell_j)} g_m!} \prod_{k=1}^{j} \binom{\ell - \sum_{i=1}^{k-1} \ell_i}{\ell_k}^2 \frac{(\ell_k!)^2}{2\ell_k} \tag{4.32}$$

The simplification of the factors $(\ell_k!)^2$ yields the expression stated by this lemma. □

Lemma 4.4. *Consider bipartite graphs defined as follows: ℓ left vertices, ℓ right vertices, all left vertices have degree 2 except one vertex of degree 1, all right vertices have degree 2 except one vertex of degree 1, and finally no length-2 cycles are allowed. For $\ell \geq 3$, the total number y_ℓ of such bipartite graphs is*

$$y_\ell = \ell^2 \cdot \left((2\ell - 1) \cdot x_{\ell-1} + (\ell - 1)^2 \cdot x_{\ell-2}\right), \tag{4.33}$$

where x_ℓ is determined via Lemma 4.3 and $x_1 = 0$.

Proof. Let the first $\ell - 1$ left vertices and the first $\ell - 1$ right vertices be of degree 2. There exists two ways to complete this bipartite graph such that the two remaining vertices have degree 1.

- Each of the $x_{\ell-1}$ sub-graphs has $2(\ell-1)$ edges. Break one edge into two edges and connect them to the remaining left and right vertices, the number of such graphs is $2(\ell-1)x_{\ell-1}$. Another set of $x_{\ell-1}$ bipartite graphs is built by directly connecting the last two vertices together without breaking any edge in the upper sub-graph. Now, we find $2(\ell-1)x_{\ell-1} + x_{\ell-1} = (2\ell-1)x_{\ell-1}$ bipartite graphs.

- Fix a vertex among the $\ell-1$ upper left vertices and fix one among the $\ell-1$ upper right vertices ($(\ell-1)^2$ choices). Consider a length-2 cycle including these two vertices. One edge of this cycle can be broken into two edges and then attached to the degree-1 vertices at the bottom. The remaining $\ell-2$ left and right vertices may involve $x_{\ell-2}$ sub-graphs. Consequently, the number of graphs in this second case is $(\ell-1)^2 x_{\ell-2}$.

The total number of bipartite graphs enumerated in the above cases is

$$(2\ell - 1)x_{\ell-1} + (\ell - 1)^2 x_{\ell-2}. \tag{4.34}$$

Finally, the degree-1 left vertex has ℓ choices and so has the degree-1 right vertex. The number of graphs in (4.34) should be multiplied by ℓ^2. □

We make no claims about a possible generalization of Lemma 4.3 and Lemma 4.4 to finite bipartite graphs with higher vertex degrees. As mentioned before, for general degree distributions, results on enumeration of asymptotic bipartite graphs were published by Brendan McKay and his co-authors [20] [24]. Table 4.1 shows the number of special partitions for $\ell = 2 \ldots 32$. The number of standard partitions (the partition function) can be found by a recursion resulting from the pentagonal number theorem [22]. To our knowledge, there exists no such recursion for special partitions. The number of bipartite graphs under the assumptions of Lemma 4.3 and Lemma 4.4 is found in Table 4.2 for a graph half-size up to 8. Finally, we are ready to state and prove the first theorem on stopping sets enumeration.

1, 1, 2, 2, 4, 4, 7, 8, 12, 14, 21, 24, 34, 41, 55, 66, 88, 105, 137, 165, 210, 253, 320, 383, 478, 574, 708, 847, 1039, 1238, 1507

Table 4.1: Sequence of the number of special partitions of the integer ℓ, for $\ell = 2 \ldots 32$. Special partitions are described in Definition 4.10. The sequence for standard partitions can be found in [97].

ℓ	2	3	4	5	6	7	8
x_ℓ	1	6	90	2040	67950	3110940	187530840
y_ℓ	0	45	816	22650	888840	46882710	3199593600

Table 4.2: Number of bipartite graphs not including length-2 cycles from Lemma 4.3 and Lemma 4.4.

In the sequel, the open interval between two real numbers a and b will be denoted $]a, b[$,

$$]a, b[\ = \ \{x \in \mathbb{R} : a < x < b\}.$$

Theorem 4.2. *Let C_P be a product code $[n_1, k_1, d_1]_q \otimes [n_2, k_2, d_2]_q$ built from row and column MDS component codes, where the alphabet size q is greater than $\max(n_1, n_2)$. Let τ_w be the number of stopping sets of size w. We write $\tau_w = \tau^a + \tau^b$, where τ^a counts obvious stopping sets and τ^b counts non-obvious stopping sets. Under (type-II) iterative algebraic decoding and for $d_1 = d_2 = d \geq 2$, stopping sets are characterized as follows:*

- *For* $w < d^2$,
$$\tau^a = \tau^b = 0.$$
- *For* $w = d^2$,
$$\tau^a = \binom{n_1}{d}\binom{n_2}{d}, \quad \tau^b = 0.$$
- *For* $w \in]d^2, d(d+1)[$,
$$\tau^a = \tau^b = 0.$$
- *For* $w = d(d+1)$,
$$\tau^a = \binom{n_1}{d}\binom{n_2}{d+1} + \binom{n_1}{d+1}\binom{n_2}{d},$$
$$\tau^b = (d+1)!\binom{n_1}{d+1}\binom{n_2}{d+1}.$$
- *For* $w \in\ ]d(d+1), d(d+2)[$.
Let us write $w = d^2 + d + \lambda$, *where* $\lambda \in [1, d-1]$.
$$\tau^a = 0,$$
$$\tau^b = (d+1-\lambda)!\binom{d+1}{\lambda}^2\binom{n_1}{d+1}\binom{n_2}{d+1}.$$
- *For* $w = d(d+2)$,
$$\tau^a = \binom{n_1}{d}\binom{n_2}{d+2} + \binom{n_1}{d+2}\binom{n_2}{d},$$
$$\tau^b = (d+1)^2\binom{n_1}{d+1}\binom{n_2}{d+1} + \sum_{2r_0+r_1=d}\binom{d+1}{r_0}\binom{d+1-r_0}{r_1}\frac{(d+2)!}{2^{r_2}}\left[\binom{n_1}{d+1}\binom{n_2}{d+2} + \binom{n_1}{d+2}\binom{n_2}{d+1}\right] + x_{d+2}\binom{n_1}{d+2}\binom{n_2}{d+2},$$
where $\sum_{2r_0+r_1=d}$ *is a summation over* r_0 *and* r_1, *both being non-negative and satisfying* $2r_0+r_1 = d$, $r_2 = d+1-r_0-r_1$, *and* x_{d+2} *is determined from Lemma 4.3.*
- *For* $w = (d+1)(d+1)$
$$\tau^a = \binom{n_1}{d+1}\binom{n_2}{d+1},$$
$$\tau^b = \sum_{2r_0+r_1=d+1}\binom{d+1}{r_0}\binom{d+1-r_0}{r_1}\frac{(d+2)!}{2^{r_0}}\left[\binom{n_1}{d+1}\binom{n_2}{d+2} + \binom{n_1}{d+2}\binom{n_2}{d+1}\right] + y_{d+2}\binom{n_1}{d+2}\binom{n_2}{d+2},$$

where y_{d+2} is determined from Lemma 4.4.

Proof. For w satisfying $d^2 \leq w \leq (d+1)^2$, the admissible size of $\mathcal{R}(\mathcal{S})$ varies from d^2 up to $(d+2)^2$ as given by Lemma 4.1. All cases stated in the theorem shall use the following sequence of $\mathcal{R}(\mathcal{S})$ listed in the order of increasing size $\ell_1\ell_2$: d^2, $d(d+1)$, $d(d+2)$, $(d+1)^2$, $(d+1)(d+2)$, and $(d+2)^2$. For these rectangular supports, the stopping set weight also has six cases to be considered, where w takes the following values (or ranges) in increasing order: $w = d^2$, $w \in]d^2, d(d+1)[$, $w = d(d+1)$, $w \in]d(d+1), d(d+2)[$, $w = d(d+2)$, and $w = (d+1)^2$.

- The case $w < d^2$.
  Consider a stopping set of size $w < d^2$. Its rectangular support $\mathcal{R}(\mathcal{S})$ has size $\ell_1\ell_2 \geq w$. All columns should have a weight greater than or equal to d, we find that $w \geq d\ell_2$. Similarly, all rows must have a weight greater than or equal to d, then $w \geq d\ell_1$. By combining the two inequalities, we find $w^2 \geq d^2\ell_1\ell_2 \geq d^2 w$, so we get $w \geq d^2$ which is a contradiction unless these stopping sets do not exist, i.e. $\tau_w = 0$ for $w < d^2$ under type II iterative decoding.

- The case $w = d^2$.
  We use similar inequalities as in the previous case. We have $w = d^2 \geq d\ell_2$ because column decoding must fail. We obtain $\ell_2 \leq d$. In a symmetric way, $w = d^2 \geq d\ell_1$ because row decoding must fail. We obtain $\ell_1 \leq d$. But $\mathcal{R}(\mathcal{S})$ cannot be smaller than $\mathcal{S}$, i.e. we get $\ell_1 = d$ and $\ell_2 = d$. We just proved that all stopping set of size d^2 are obvious. Their number is given by choosing d rows out of n_1 and d columns out of n_2.

- The case $d^2 < w < d(d+1)$.
  Given that $\ell_1\ell_2 \geq w > d^2$, we get $\ell_1 \geq d$ and $\ell_2 \geq d$ since the support $\mathcal{R}(\mathcal{S})$ is larger than a $d \times d$ rectangle, the latter being the smallest stopping set as proven in the previous case. Take $\ell_1 = d$, then $\ell_2 \geq d+1$ because $w > d^2$. The weight of each column must be at least d giving us $w \geq d\ell_2 \geq d(d+1)$, which is a contradiction unless $\tau_w = 0$. For $\ell_1 > d$, the same arguments hold.

- The case $w = d(d+1)$.
  - The smallest $\mathcal{R}(\mathcal{S})$ is $d \times (d+1)$ or $(d+1) \times d$. According to Lemma 4.2, we have $\beta = 0$. All these stopping sets are obvious. Their number is
    $$\binom{n_1}{d}\binom{n_2}{d+1} + \binom{n_1}{d+1}\binom{n_2}{d}.$$
  - $\mathcal{R}(\mathcal{S})$ has size $d(d+2)$. Each column must have at least d erasures. Then $\mathcal{S}$ can only be obvious with weight $d(d+2)$ which contradicts $w = d(d+1)$. Hence, this size of rectangular support yields no stopping sets, $\tau_w = 0$ in this sub-case.

  - $\mathcal{R}(\mathcal{S})$ has size $(d+1)(d+1)$. Let β be the number of zeros in $\mathcal{R}(\mathcal{S})$, $\beta = (d+1)^2 - w = d+1$. All these stopping sets are found by considering the $(d+1)!$ permutations where a unique 0 is placed per row and per column. Then, the binomial coefficient must be multiplied by $(d+1)!$ which yields the τ^b announced in the theorem for $w = d(d+1)$.
  - $\mathcal{R}(\mathcal{S})$ has size $(d+1)(d+2)$. The number of zeros is $\beta = (d+1)(d+2) - w = 2d+2$. Then $\beta > d+2 = (\ell_1 - d)\ell_2$ which contradicts Lemma 4.2. We get $\tau_w = 0$ in this sub-case. The same arguments are valid for larger rectangles.

- The case $d(d+1) < w < d(d+2)$.
  Let us write $w = d^2+d+\lambda$, where $\lambda \in [1, d-1]$. We consider below three sub-cases corresponding to admissible sizes of $\mathcal{R}(\mathcal{S})$.

  - The smallest $\mathcal{R}(\mathcal{S})$ is $d \times (d+2)$ or $(d+2) \times d$. Take the rectangle of size $d \times (d+2)$. Each column must have at least d erasures. Then $\mathcal{S}$ can only be obvious with weight $d(d+2)$ which is outside the range for w in this case. Hence, this size of the rectangular support yields no stopping sets, $\tau_w = 0$ in this sub-case.
  - $\mathcal{R}(\mathcal{S})$ has size $(d+1) \times (d+1)$. The number of zeros is $\beta = (d+1)^2 - w = d+1-\lambda$, where $\beta \in [2, d]$.
    Put the zeros in $\mathcal{R}(\mathcal{S})$ not exceeding one per column and not exceeding one per row. The enumeration of these stopping sets is given by selecting the β rows and the β columns, then filling all $\beta \times \beta$ permutation matrices in the zero positions. Hence, given that $\binom{d+1}{\beta} = \binom{d+1}{\lambda}$, we get for this sub-case

$$\tau_w = \beta! \binom{d+1}{\lambda}^2 \binom{n_1}{d+1} \binom{n_2}{d+1}.$$

    All corresponding stopping sets are not obvious (the rank is greater than 1).
  - $\mathcal{R}(\mathcal{S})$ has size $(d+1)(d+2)$. The number of zeros is $\beta = (d+1)(d+2) - w = 2d+2-\lambda \in [d+3, 2d+1]$. Then $\beta > d+2 = (\ell_1 - d)\ell_2$ which contradicts Lemma 4.2. In a similar way, it can be proven that $\tau_w = 0$ in the sub-case $\mathcal{R}(\mathcal{S})$ with size $(d+2)^2$.

- The case $w = d(d+2)$.
  The admissible rectangular support can have four sizes: $d(d+2)$, $(d+1)(d+1)$, $(d+1)(d+2)$, and $(d+2)(d+2)$.

  - $\mathcal{R}(\mathcal{S})$ has size $d(d+2)$. According to Lemma 4.2, we have $\beta = 0$. All these stopping sets are obvious. Their number is

$$\binom{n_1}{d}\binom{n_2}{d+2} + \binom{n_1}{d+2}\binom{n_2}{d}.$$

– $\mathcal{R}(\mathcal{S})$ has size $(d+1)(d+1)$. We have $\beta = 1$. The number of these stopping sets is

$$(d+1)^2 \binom{n_1}{d+1} \binom{n_2}{d+1}.$$

– $\mathcal{R}(\mathcal{S})$ has size $(d+1)(d+2)$. The number of zeros is $\beta = d+2$. Each column must have a unique zero and each row cannot have more than two zeros. Let r_i be the number of rows containing i zeros, $i = 0, 1, 2$. Then $r_0 + r_1 + r_2 = d+1$ and $\beta = 2r_2 + r_1$, so the constraint is $2r_0 + r_1 = d$. Given a stopping set satisfying this constraint, a permutation can be applied on the $(d+2)$ columns to create another stopping set. But a row with two zeros creates two identical columns, so the number of stopping sets should be divided by 2^{r_2}, where $r_2 = r_0 + r_1 - d - 1$. The number of stopping sets in this sub-case is

$$\sum_{2r_0+r_1=d} \binom{d+1}{r_0} \binom{d+1-r_0}{r_1} \frac{(d+2)!}{2^{r_2}} \left[\binom{n_1}{d+1} \binom{n_2}{d+2} + \binom{n_1}{d+2} \binom{n_2}{d+1} \right].$$

– $\mathcal{R}(\mathcal{S})$ has size $(d+2)(d+2)$. We have $\beta = 2d+4$ reaching the upper bound in Lemma 4.2. $\mathcal{R}(\mathcal{S})$ must have two zeros in each column and two zeros in each row. A first group of these stopping sets can be enumerated by building $\mathcal{R}(\mathcal{S})$ with two zero length-$(d+2)$ diagonals (to be folded if not the main diagonal) and then applying all row and column permutations. This generates all Hamiltonian bipartite graphs with $d+2$ left vertices and $d+2$ right vertices, their number is

$$\frac{((d+2)!)^2}{2(d+2)},$$

as known from Lemma 4.3. In fact, the full exact enumeration of stopping sets in this case is already made by Lemma 4.3 and its proof, just take $\ell = d+2$. Then, in this sub-case, the number of stopping sets is given by

$$x_{d+2} \binom{n_1}{d+2} \binom{n_2}{d+2}.$$

- The case $w = (d+1)(d+1)$.
The admissible rectangular support can have three possible sizes $(d+1)(d+1)$, $(d+1)(d+2)$, and $(d+2)(d+2)$.

– $\mathcal{R}(\mathcal{S})$ has size $(d+1)(d+1)$. We have $\beta = 0$, i.e. $\mathcal{R}(\mathcal{S}) = \mathcal{S}$. The number of these obvious stopping sets is

$$\binom{n_1}{d+1} \binom{n_2}{d+1}.$$

– $\mathcal{R}(\mathcal{S})$ has size $(d+1)(d+2)$. We have $\beta = d+1$. A column of $\mathcal{R}(\mathcal{S})$ should contain at most one zero and a row should contain at most two zeros. Let r_i be the number of rows containing i zeros, $i = 0, 1, 2$. Then $r_0 + r_1 + r_2 = d+1$ and $\beta = 2r_2 + r_1$, so the constraint is $2r_0 + r_1 = d+1$. Given a stopping set satisfying this constraint, a permutation can be applied on the $(d+2)$ columns to create another stopping set. The number of stopping sets in this sub-case is

$$\sum_{2r_0+r_1=d+1} \binom{d+1}{r_0}\binom{d+1-r_0}{r_1}\frac{(d+2)!}{2^{r_2}}\left[\binom{n_1}{d+1}\binom{n_2}{d+2}+\binom{n_1}{d+2}\binom{n_2}{d+1}\right],$$

where $r_2 = r_0$.

– $\mathcal{R}(\mathcal{S})$ has size $(d+2)(d+2)$. $\beta = 2d+3$ which is less than the upper bound in Lemma 4.2. These stopping sets are equivalent to bipartite graphs considered in Lemma 4.4. Then, in this sub-case, the number of stopping sets is given by

$$y_{d+2}\binom{n_1}{d+2}\binom{n_2}{d+2}.$$

□

From the proof of Theorem 4.2, in the case $w = d(d+2)$ with a $(d+1) \times (d+2)$ rectangular support, the enumeration of stopping sets is directly converted into enumeration of trivial bipartite graphs defined by: a- ℓ left vertices and a left degree 0, 1, or 2, and b- $\ell + 1$ right vertices all of degree 1. Similarly, the proof for the case $w = d(d+2)$ with a $(d+1) \times (d+2)$ rectangular support is directly related to the enumeration of bipartite graphs with one edge less.

Theorem 4.3. *Let C_P be a product code $[n_1, k_1, d_1]_q \otimes [n_2, k_2, d_2]_q$ built from row and column MDS components, where the alphabet size q is greater than $\max(n_1, n_2)$. Let τ_w be the number of stopping sets of Hamming weight w. We write $\tau_w = \tau^a + \tau^b$, where τ^a counts obvious stopping sets and τ^b counts non-obvious stopping sets. It is assumed that $2 < d_1 < d_2 < 3d_1 - 1$ or $2 = d_1 < d_2 < 4d_1 - 1$. Under iterative algebraic decoding, stopping sets are characterized as follows.*

- *For $w < d_1 d_2$,*
$$\tau^a = \tau^b = 0.$$
- *For $w = d_1 d_2$,*
$$\tau^a = \binom{n_1}{d_1}\binom{n_2}{d_2}, \quad \tau^b = 0.$$
- *For $w \in]d_1 d_2, d_1(d_2+1)[$,*
$$\tau^a = \tau^b = 0.$$

- *For* $w = d_1(d_2+1)$,
$$\tau^a = \binom{n_1}{d_1}\binom{n_2}{d_2+1}, \quad \tau^b = 0.$$

For larger weights, the enumeration of stopping sets distinguishes three cases: A, B, and C.

Case A: $d_2 < 2d_1$**.**

- *For* $w \in]d_1(d_2+1), (d_1+1)d_2[$.
$$\tau^a = \tau^b = 0.$$

- *For* $w = (d_1+1)d_2$.
$$\begin{aligned}\tau^a &= \binom{n_1}{d_1+1}\binom{n_2}{d_2},\\ \tau^b &= (d_1+1)!\binom{d_2+1}{d_2-d_1}\binom{n_1}{d_1+1}\binom{n_2}{d_2+1}.\end{aligned}$$

- *For* $w \in](d_1+1)d_2, d_1(d_2+2)[$, *write* $w = (d_1+1)d_2 + \lambda$.
$$\tau^b = \mathbb{1}_{\{d_2<2d_1-1\}} \times (d_1+1-\lambda)!\binom{d_1+1}{\lambda}\binom{d_2+1}{d_1+1-\lambda}\binom{n_1}{d_1+1}\binom{n_2}{d_2+1}.$$

- *For* $w = d_1(d_2+2)$.
$$\begin{aligned}\tau^a &= \binom{n_1}{d_1}\binom{n_2}{d_2+2},\\ \tau^b &= (d_2-d_1+1)!\binom{d_1+1}{d_2-d_1+1}\binom{d_2+1}{d_2-d_1+1}\binom{n_1}{d_1+1}\binom{n_2}{d_2+1}\\ &\quad + \sum_{2r_0+r_1=2d_1-d_2}\binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1}\frac{(d_2+2)!}{2^{r_0+d_2-d_1+1}}\binom{n_1}{d_1+1}\binom{n_2}{d_2+2}.\end{aligned}$$

- *For* $w \in]d_1(d_2+2), (d_1+1)(d_2+1)[$, *write* $w = d_1(d_2+2)+\lambda$.
$$\begin{aligned}\tau^b &= (d_2-d_1+1-\lambda)!\binom{d_1+1}{2d_1-d_2+\lambda}\binom{d_2+1}{d_1+\lambda}\binom{n_1}{d_1+1}\binom{n_2}{d_2+1}\\ &\quad + \sum_{2r_0+r_1=2d_1-d_2+\lambda}\binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1}\frac{(d_2+2)!}{2^{r_2}\lambda!}\binom{n_1}{d_1+1}\binom{n_2}{d_2+2},\end{aligned}$$

where $r_2 = r_0 + \lambda - d_1 - 1$.

- *For* $w = (d_1+1)(d_2+1)$.

$$\begin{aligned}
\tau^a &= \binom{n_1}{d_1+1}\binom{n_2}{d_2+1} + \mathbb{1}_{\{d_2=2d_1-1\}}\binom{n_1}{d_1}\binom{n_2}{d_2+3} + \mathbb{1}_{\{d_2=d_1+1\}}\binom{n_1}{d_1+2}\binom{n_2}{d_2},\\
\tau^b &= \sum_{2r_0+r_1=d_1+1}\binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1}\frac{(d_2+2)!}{2^{r_0}(d_2-d_1+1)!}\binom{n_1}{d_1+1}\binom{n_2}{d_2+2}\\
&\quad + \mathbb{1}_{\{d_2=d_1+1\}}\times\sum_{2r_0+r_1=d_2+1}\binom{d_2+1}{r_0}\binom{d_2+1-r_0}{r_1}\frac{(d_1+2)!}{2^{r_0}}\binom{n_1}{d_1+2}\binom{n_2}{d_2+1}\\
&\quad + \mathbb{1}_{\{d_2=2d_1-1\}}\times\sum_{3r_0+2r_1+r_2=d_1+1}\binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1}\times\\
&\qquad\qquad \binom{d_1+1-r_0-r_1}{r_2}\frac{(d_2+3)!}{2^{r_2}6^{r_3}}\binom{n_1}{d_1+1}\binom{n_2}{d_2+3}\\
&\quad + \mathbb{1}_{\{d_2=d_1+1\}}\left((d_2+2)x_{d_1+2} + \frac{(d_2+2)y_{d_1+2}}{2}\right)\binom{n_1}{d_1+2}\binom{n_2}{d_2+2}\\
&\quad + \mathbb{1}_{\{d_1=2,d_2=3\}}\times 1860\binom{n_1}{d_1+2}\binom{n_2}{d_2+3}.
\end{aligned}$$

  where $r_3 = d_1+1-r_0-r_1-r_2$, *and* x_{d_1+2} *and* y_{d_1+2} *are determined from Lemma 4.3 and Lemma 4.4 respectively.*

Case B: $d_2 = 2d_1$**.**

- *For* $w \in]d_1(d_2+1), (d_1+1)d_2[$.

$$\tau^a = \tau^b = 0.$$

- *For* $w = (d_1+1)d_2 = d_1(d_2+2)$.

$$\begin{aligned}
\tau^a &= \binom{n_1}{d_1+1}\binom{n_2}{d_2} + \binom{n_1}{d_1}\binom{n_2}{d_2+2},\\
\tau^b &= (d_1+1)!\binom{d_2+1}{d_1+1}\binom{n_1}{d_1+1}\binom{n_2}{d_2+1} + \frac{(d_2+2)!}{2^{d_1+1}}\binom{n_1}{d_1+1}\binom{n_2}{d_2+2}.
\end{aligned}$$

- *For* $w \in](d_1+1)d_2, d_1(d_2+3)[$, *write* $w = (d_1+1)d_2 + \lambda$.

$$\begin{aligned}
\tau^b &= (d_1+1-\lambda)!\binom{d_1+1}{\lambda}\binom{d_2+1}{d_1+\lambda}\binom{n_1}{d_1+1}\binom{n_2}{d_2+1}\\
&\quad + \sum_{2r_0+r_1=\lambda}\binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1}\frac{(d_2+2)!}{2^{r_2}\lambda!}\binom{n_1}{d_1+1}\binom{n_2}{d_2+2},
\end{aligned}$$

  where $r_2 = d_1+1-r_0-r_1 = d_1+1+r_0-\lambda$.

- *For* $w = d_1(d_2+3)$.

$$\tau^a = \binom{n_1}{d_1}\binom{n_2}{d_2+3},$$
$$\tau^b = (d_1+1)(d_2+1)\binom{n_1}{d_1+1}\binom{n_2}{d_2+1} + \sum_{2r_0+r_1=d_1}\binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1}\frac{(d_2+2)!}{2^{r_2}d_1!}\binom{n_1}{d_1+1}\binom{n_2}{d_2+2} + \sum_{3r_0+2r_1+r_2=d_1}\binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1}\binom{d_1+1-r_0-r_1}{r_2}\frac{(d_2+3)!}{2^{r_2}6^{r_3}}\binom{n_1}{d_1+1}\binom{n_2}{d_2+3},$$

  where $r_3 = d_1+1-r_0-r_1-r_2 = 2r_0+r_1+1$.

- *For* $w = (d_1+1)(d_2+1)$.

$$\tau^a = \binom{n_1}{d_1+1}\binom{n_2}{d_2+1},$$
$$\tau^b = \sum_{2r_0+r_1=d_1+1}\binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1}\frac{(d_2+2)!}{2^{r_0}(d_1+1)!}\binom{n_1}{d_1+1}\binom{n_2}{d_2+2} + \sum_{3r_0+2r_1+r_2=d_1+1}\binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1}\binom{d_1+1-r_0-r_1}{r_2}\times \frac{(d_2+3)!}{2^{r_2}6^{2r_0+r_1}}\binom{n_1}{d_1+1}\binom{n_2}{d_2+3}.$$

Case C: $4 < 2d_1 < d_2 < 3d_1 - 1$
or $4 = 2d_1 < d_2 < 4d_1 - 1$**.**

- *For* $w \in]d_1(d_2+1), d_1(d_2+2)[$.

$$\tau^a = \tau^b = 0.$$

- *For* $w = d_1(d_2+2)$.

$$\tau^a = \binom{n_1}{d_1}\binom{n_2}{d_2+2}.$$

- *For* $w \in]d_1(d_2+2), (d_1+1)d_2[$.

$$\tau^a = \tau^b = 0.$$

- *For* $w = (d_1+1)d_2$.

$$\tau^a = \binom{n_1}{d_1+1}\binom{n_2}{d_2},$$
$$\tau^b = (d_1+1)!\binom{d_2+1}{d_1+1}\binom{n_1}{d_1+1}\binom{n_2}{d_2+1} + \frac{(d_2+2)!}{2^{d_1+1}(d_2-2d_1)!}\binom{n_1}{d_1+1}\binom{n_2}{d_2+2} + \mathbb{1}_{\{d_1=2,d_2=6\}} \times 1680\binom{n_1}{d_1+1}\binom{n_2}{d_2+3}.$$

- *For* $w \in](d_1+1)d_2, d_1(d_2+3)[$, *write* $w = (d_1+1)d_2 + \lambda$.

$$\tau^b = \mathbb{1}_{\{d_1>2\}} \times \left[(d_1+1-\lambda)!\binom{d_2+1}{d_1+1-\lambda}\binom{d_1+1}{\lambda}\binom{n_1}{d_1+1}\binom{n_2}{d_2+1} + \sum_{2r_0+r_1=\lambda}\binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1}\frac{(d_2+2)!}{2^{r_2}(d_2-2d_1+\lambda)!}\binom{n_1}{d_1+1}\binom{n_2}{d_2+2}\right].$$

where $r_2 = d_1+1+r_0-\lambda$.

- *For* $w = d_1(d_2+3)$.

$$\tau^a = \binom{n_1}{d_1}\binom{n_2}{d_2+3},$$
$$\tau^b = \mathbb{1}_{\{d_1=2,d_2=6\}\bigcup\{d_1>2\}} \times \left[(d_2-2d_1+1)!\binom{d_1+1}{3d_1-d_2}\binom{d_2+1}{2d_1}\binom{n_1}{d_1+1}\binom{n_2}{d_2+1} + \sum_{2r_0+r_1=3d_1-d_2}\binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1}\frac{(d_2+2)!}{2^{r_2}d_1!}\binom{n_1}{d_1+1}\binom{n_2}{d_2+2} + \sum_{3r_0+2r_1+r_2=3d_1-d_2}\binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1}\binom{d_1+1-r_0-r_1}{r_2}\frac{(d_2+3)!}{2^{r_2}6^{r_3}}\times \binom{n_1}{d_1+1}\binom{n_2}{d_2+3}\right],$$

where $r_2 = d_2-2d_1+1+r_0$ *and* $r_3 = d_1+1-r_0-r_1-r_2$.

- *For* $w = d_1(d_2+4)$.

$$\tau^a = \mathbb{1}_{\{d_1=2\}} \times \binom{n_1}{d_1}\binom{n_2}{d_2+4}.$$

- *For* $w \in]d_1(d_2+3), (d_1+1)(d_2+1)[$, *write* $w = d_1(d_2+3)+\lambda$.

$$\tau^b = (d_2 - 2d_1 + 1 - \lambda)!\binom{d_1+1}{3d_1-d_2+\lambda}\binom{d_2+1}{2d_1+\lambda}\binom{n_1}{d_1+1}\binom{n_2}{d_2+1}$$
$$+ \sum_{2r_0+r_1=3d_1-d_2+\lambda} \binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1}\frac{(d_2+2)!}{2^{r_2}(d_1+\lambda)!}\binom{n_1}{d_1+1}\binom{n_2}{d_2+2}$$
$$+ \sum_{3r_0+2r_1+r_2=3d_1-d_2+\lambda} \binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1}\binom{d_1+1-r_0-r_1}{r_2}\times$$
$$\frac{(d_2+3)!}{2^{r_2}6^{r_3}\lambda!}\binom{n_1}{d_1+1}\binom{n_2}{d_2+3}$$
$$+ \mathbb{1}_{\{d_1=2,d_2=6\}} \times 22050\binom{n_1}{d_1+1}\binom{n_2}{d_2+4},$$

where $r_2 = d_1 + 1 - r_0 - r_1$ *and* $r_3 = d_1 + 1 - r_0 - r_1 - r_2$.

- *For* $w = (d_1 + 1)(d_2 + 1)$.

$$\tau^a = \binom{n_1}{d_1+1}\binom{n_2}{d_2+1},$$
$$\tau^b = (d_1+1)!\binom{d_2+2}{d_1+1}\binom{n_1}{d_1+1}\binom{n_2}{d_2+2}$$
$$+ \sum_{3r_0+2r_1+r_2=d_1+1} \binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1}\binom{d_1+1-r_0-r_1}{r_2}\times$$
$$\frac{(d_2+3)!}{2^{r_2}6^{r_3}(d_2-2d_1+1)!}\binom{n_1}{d_1+1}\binom{n_2}{d_2+3}$$
$$+ \mathbb{1}_{\{d_1=2,d_2=5\}} \times 11130\binom{n_1}{d_1+1}\binom{n_2}{d_2+4}$$
$$+ \mathbb{1}_{\{d_1=2,d_2=6\}} \times 111300\binom{n_1}{d_1+1}\binom{n_2}{d_2+4},$$

where $r_3 = d_1 + 1 - r_0 - r_1 - r_2$.

The detailed proof of Theorem 4.3 is found in Appendix A. From a stopping set perspective, both theorems 4.2&4.3 match Tolhuizen's results on weight distribution for a weight less than $d_1d_2+d_2$ [105]. Our theorems found stopping sets that are only obvious for w in the range $[d_1d_2, d_1d_2+d_2[$. For any weight w, there exists an equivalence in support between codewords and obvious stopping sets (thanks to Proposition 4.3). Trivial lower and upper bounds of the number of obvious weight-w product code codewords is

$$(q-1)\tau_w \leq A_w \leq \mathbb{1}_{\{\tau_w \neq 0\}} A_w.$$

For non-obvious stopping sets and non-obvious codewords, establishing a clear relationship is still an open problem. This is directly related to solving the weight enumeration beyond $d_1 d_2 + \max(d_1, d_2)$. In the special case $d_1 = d_2 = d$, Sendrier gave upper bounds of the number of erasure patterns for a weight up to $d^2 + 2d - 1$ [95].

4.4 Edge coloring algorithm under constraints

In section 4.2, we described graph representations of product codes and we introduced the root order $\rho(e)$ of an edge with respect to its color $\phi(e)$. Our objective is to find a coloring ϕ such that the maximum diversity order is reached under block erasures. The notion of root order in Def. (4.7) is for double diversity ($L = 2$) because it indirectly assumes that all symbols of one color out of M can be erased by the channel. Given the Singleton bound tradeoff stated in (4.7), double diversity is sufficient in distributed storage applications where the required coding rate should be sufficiently high. Def. (4.7) may be generalized to take into account two or more erased colors, e.g. see Figure 11 in [17] for $L = 3$ with $M = 3$ colors where an information symbol is protected by multiple rootcheck nodes. In this chapter, we restrict both Def. (4.7) and the design in this section to a double-diversity product code. This double diversity on a block-erasure channel is achieved if all stopping sets, as defined and counted in the previous section, can be colored in a way such that at least two distinct colors are found within the symbols of a stopping set (valid for both $\mathcal{G}$ and $\mathcal{G}^c$). This task is intractable. Imagine an edge coloring ϕ designed in a way to guarantee that all weight-w stopping sets include at least two colors. This task is already very hard (or almost impossible) for a fixed w. There is no coloring design tool for non-trivial product codes to ensure that all stopping sets of all weights incorporate at least two distinct colors.

4.4.1 Hand-made edge coloring and its limitations

The aim of this section is to give more insight on designing edge coloring, before introducing the differential evolution algorithm.

The compact graph $\mathcal{G}^c$ makes the design much simpler, as we saw in Section 4.3.3. The number of super-edges with the same color is N^c/M. We also know from (4.11)-(4.13) that the size, height, and width of G^c are directly related to the component and total coding rates.

Lemma 4.5. *Let $C_P = C_1 \otimes C_2$ be a product code with a column component $C_1[n_1, k_1]_q$ and a row component $C_2[n_2, k_2]_q$ whose coding rates are $R_1 = k_1/n_1$ and $R_2 = k_2/n_2$ respectively. Assume that $n_i - k_i$ divides n_i, for $i = 1, 2$, and assume that M divides N^c. G^c admits an edge coloring ϕ such that $\rho_{max}(\phi) = 1$ if the coding rates satisfy*

$$\min(R_1, R_2) \leq 1 - \frac{1}{M}. \tag{4.35}$$

Proof. Consider the $|V_1^c| \times |V_2^c|$ matrix representation of $\mathcal{G}^c$. A sufficient condition to get $\rho_{max}(\phi) = 1$ is to assign the N^c/M edges having the same color to a single row or a single column. The sufficient condition for $\rho_{max}(\phi) = 1$ is expressed as $N^c/M \leq \max(n_1/(n_1 - k_1), n_2/(n_2 - k_2))$, the max let us select the longest item among a row or a column. Recall also that $|V_i^c| = n_i/(n_i - k_i)$. Using (4.2), the sufficient condition becomes $n_1 n_2 \leq M \cdot \max(n_1(n_2 - k_2), n_2(n_1 - k_1))$. Divide by $n_1 n_2$ to get the inequality announced in the Lemma statement. □

When the palette has $M = 4$ colors, the sufficient condition in Lemma 4.5 is written as $\min(R_1, R_2) \leq 3/4$. In order to achieve the block-fading Singleton bound for $M = 4$, we should take $R_1 = 3/4$ and $R_2 = 1$, i.e. the product code degenerates to a single component code. It is possible to approach $R = 3/4$ by keeping $R_1 = 3/4$ and letting $R_2 = \frac{n_2 - 1}{n_2}$ be very close to 1. In this case, the row code C_2 is a single-parity check code over F_q. The product code is very unbalanced. An example of such an unbalanced product code is

$$C_P = [12, 9, 4]_q \otimes [14, 13, 2]_q.$$

From the proof of Lemma 4.5, the edge coloring of $\mathcal{G}^c$ satisfying $\rho_{max} = 1$ is given by the following 4×14 matrix:

$$\begin{bmatrix} R & R & R & \dots & R & R \\ G & G & G & \dots & G & G \\ B & B & B & \dots & B & B \\ Y & Y & Y & \dots & Y & Y \end{bmatrix}, \tag{4.36}$$

where the colors $\phi(e) = 1, 2, 3, 4$ are replaced by the four letters 'R', 'G', 'B', and 'Y'. The rate of $[12, 9, 4]_q \otimes [14, 13, 2]_q$ is comparable to the rate of $[12, 10, 3]_q^{\otimes 2}$, $R \approx 0.69$ but it is sill far from reaching three quarters as the product code $[14, 12, 3]_q \otimes [16, 14, 3]_q$. Of course, if the practical constraints allow for it, it is possible to consider an extremely unbalanced code such as $[12, 9, 4]_q \otimes [100, 99, 2]_q$!

Let us build balanced product codes by relaxing the constraint $\rho_{max} = 1$. We may authorize a ρ_{max} greater than 1 but not too large in order to limit the number of decoding iterations. On the other hand, the double diversity condition on the edge coloring is maintained. Firstly, let us find a hand-made edge coloring for the $[12, 10, 3]_q^{\otimes 2}$ product code with $M = 4$ colors. $\mathcal{G}^c$ has 6 left supernodes, 6 right supernodes, and a total of 36 edges. Each color is used $N^c/M = 9$ times. The hint is to place a color on the rows of the matrix representation of $\mathcal{G}^c$, row by row from the top to the bottom in a way that avoids stopping sets. The smallest stopping set is the 2×2 square. Other non-obvious stopping sets may not be visible without a tedious row-column decoding which is equivalent to determining the root order of all edges. We start with the first

color 'R' and use the following number of letters per row:

$$\begin{bmatrix} R & R & R & G & B & Y \\ R & & & R & & \\ R & & & & & \\ R & & & & & \\ R & & & & & \\ R & & & & & \end{bmatrix}. \tag{4.37}$$

As seen above, we completed the first row with the three other colors. On the second row, we moved the second 'R' to the right to avoid a 2×2 stopping set. Next, we can start filling the second color 'G' from the third row, then the third color 'B' from the fifth row. There will be no choice for the 9 positions of 'Y'. We allow some extra permutations to avoid small stopping sets. After filling the 36 positions, we found the following hand-made edge coloring for the $[12, 10, 3]_q^{\otimes 2}$ product code:

$$\begin{bmatrix} R & R & R & G & B & Y \\ R & B & Y & R & Y & G \\ B & G & G & G & R & Y \\ Y & G & B & Y & G & R \\ R & G & B & B & B & Y \\ R & G & B & Y & Y & B \end{bmatrix}. \tag{4.38}$$

This coloring ϕ gives 24 super-edges of order 1 (96 edges in the non-compact graph $\mathcal{G}$) and $\rho_{max}(\phi) = 3$. Can we find a better ϕ? Yes, in Section 4.4.3, the DECA algorithm outputs an edge coloring with a population of 32 super-edges of order 1 (128 edges in the non-compact graph $\mathcal{G}$) and reaching $\rho_{max}(\phi) = 2$ only.

In a similar way, we attempt to build a double-diversity coloring for a well-balanced rate-3/4 product code, e.g. the $[14, 12, 3]_q \otimes [16, 14, 3]_q$ product code where $R_1 = 6/7$, $R_2 = 7/8$, and $R = 3/4$. The compact graph $\mathcal{G}^c$ has 7 left edges and 8 right edges. For $M = 4$ colors, each color is used $N^c/M = 56/4 = 14$ times. Again, we try to avoid small obvious stopping sets like 2×2, 2×3, 3×3, etc. We start by putting five 'R' on the first row, three 'R' on the second row, two 'R' on the third row, and one 'R' on the remaining rows as follows:

$$\begin{bmatrix} R & R & R & R & R & G & B & Y \\ R & & & & & R & R & \\ R & & & & & & & R \\ R & & & & & & & \\ R & & & & & & & \\ R & & & & & & & \\ R & & & & & & & \end{bmatrix}. \tag{4.39}$$

We repeat the same number of color entries 'G' starting on the fourth row. The color 'B' starts with five entries on the seventh row. We allow some extra permutations to

avoid small stopping sets. Colors were exchanged within a row or within a column. The coloring process was tedious. Many permutations had to be applied. Some non-obvious stopping sets appeared, a computer software was used to reveal those sets (only for this task). We reached the following hand-made double-diversity edge coloring for the $[14, 12, 3]_q \otimes [16, 14, 3]_q$ product code:

$$\begin{bmatrix} Y & R & R & Y & R & G & B & R \\ R & Y & B & G & Y & R & R & B \\ B & B & B & Y & Y & R & G & R \\ R & G & G & G & G & G & B & Y \\ R & G & Y & Y & B & Y & G & G \\ G & G & R & R & B & Y & Y & Y \\ R & G & B & B & B & B & B & Y \end{bmatrix}. \tag{4.40}$$

This coloring gives 30 super-edges of order 1 in $\mathcal{G}^c$ (120 edges in the non-compact graph $\mathcal{G}$) and $\rho_{max}(\phi) = 5$. In Section 4.4.3, for the same rate-3/4 product code, the DECA algorithm outputs an edge coloring with a population of 40 super-edges of order 1 (160 edges in the non-compact graph $\mathcal{G}$) and reaching $\rho_{max}(\phi) = 3$ only.

4.4.2 The algorithm

We propose in this section an algorithm for product codes that searches for an edge coloring with a large number of root-order-1 edges (*good edges*) and achieving double diversity. The search is made in the ensemble of edge colorings $\Phi(E^c)$ of the compact graph $\mathcal{G}^c$. A necessary condition on the coding rate R to get double diversity is

$$R \leq 1 - \frac{1}{M}, \tag{4.41}$$

i.e. those satisfying inequality (4.7), where M is the color palette size. Codes attaining equality in (4.7) are referred to as MDS in the block-fading/block-erasure sense [42][17]. The main loop of our algorithm is a differential evolution loop that mutates a fraction of the population of bad edges. The algorithm will be referred to as the Differential Edge Coloring Algorithm (DECA).

The population of bad edges is defined by the following set

$$B = \{e \in E^c : \rho(e) > 1\}. \tag{4.42}$$

It should be remembered that $B = B(\phi)$ because of Def. (4.7), but ϕ is dropped here for the sake of simplifying the notations. The number of good edges is given by

$$\eta(\phi) = |E^c \setminus B| = |\{e \in E^c : \rho(e) = 1\}|. \tag{4.43}$$

Among the $|B|$ bad edges, colors of a fraction of $\aleph$ edges are modified in order to maximize $\eta(\phi)$, $\aleph \in \mathbb{N}$. The fraction $\aleph/|B|$ should be large enough to allow for a population evolution but it should stay small enough in order to limit the algorithm complexity.

The DECA algorithm proceeds as follows.

Initialization. The compact graph (V_1^c, V_2^c, E^c), the number of colors M, the differential evolution parameter $\aleph$, a maximum number of rounds $MaxIter$, and an initial edge coloring ϕ_0 are made ready as an input to DECA.

Pre-processing. Build all weak compositions of $\aleph$ with M parts, i.e. write $\aleph$ as the some of M non-negative integers,

$$\aleph = \gamma_1 + \gamma_2 + \ldots + \gamma_M, \tag{4.44}$$

the number of weak compositions being

$$\Gamma = \binom{\aleph - M + 1}{M - 1}. \tag{4.45}$$

For each weak composition, prepare the Λ permutations that permute colors among the $\aleph$ edges, the total number of these permutations is

$$\Lambda(\gamma_1, \ldots, \gamma_M) = \frac{(\gamma_1 + \ldots + \gamma_M)!}{\prod_{i=1}^{M} \gamma_i!}. \tag{4.46}$$

This pre-processing step is completed by setting a loop counter to zero.

Differential evolution loop. This looping phase of DECA includes three main steps.

- **Edge sets initialization.** Set $\phi = \phi_0$ and $\eta_{max} = 0$. Build $B = B(\phi)$ and randomly select a subset $B_\aleph$. There is a unique weak composition $(\gamma_1, \ldots, \gamma_M)$ of $\aleph$ associated to $B_\aleph$ determined by

$$\gamma_i = |\{e \in B_\aleph : \phi(e) = i\}|. \tag{4.47}$$

- **Color permutations.** For $\lambda = 1 \ldots \Lambda(\gamma_1, \ldots, \gamma_M)$, replace the image of $B_\aleph$ in the mapping ϕ by a permutation of $\phi_0(B_\aleph)$. The color permutation is denoted by π^λ. This step is a modification of the mapping ϕ_0 at the $\aleph$ bad edges, i.e. $\phi(B_\aleph) \leftarrow \pi^\lambda(\phi_0(B_\aleph))$. Record the mapping with the largest number of good edges, i.e. the edge coloring with the best $\eta(\phi)$, in ϕ_1 and update η_{max}.

- **Termination.** Increment the counter of evolution loops. Stop and output ϕ_1 if this counter reaches $MaxIter$, otherwise set $\phi_0 = \phi_1$ and go back to the edge sets initialization.

A detailed functional flowchart of DECA is drawn in Figure 4.6. The complexity of DECA is mainly due to the differential evolution loop. The complexity is proportional to $\Lambda(\gamma_1, \ldots, \gamma_M)$ per round. Hence, the number of operations in DECA behaves as

$$\Lambda \leq \Lambda_{max}(\aleph, M) = \frac{\aleph!}{((\aleph/M)!)^M}. \tag{4.48}$$

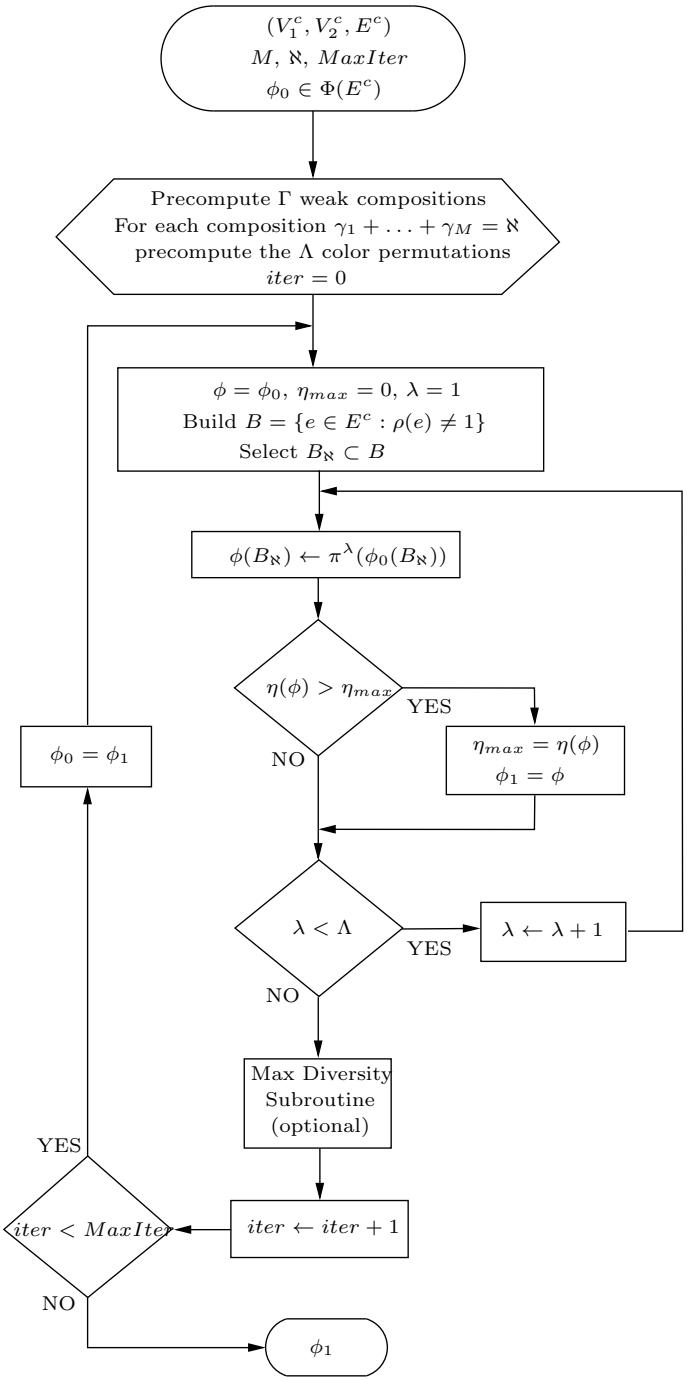


Figure 4.6: Flowchart of the edge coloring algorithm (DECA) for designing double-diversity product codes.

When $\aleph$ is not multiple of M, the denominator in the right term should be rewritten as $\prod_{i=1}^{i_0} \lfloor \aleph/M \rfloor \times \prod_{i=i_0+1}^{M} \lceil \aleph/M \rceil$, where i_0 is chosen such that the sum of all elements involved in both products is equal to $\aleph$. All Γ compositions of $\aleph$ are not considered by the algorithm. In fact, the total number of permutations for all weak compositions is

$$\sum_{j=1}^{\Gamma} \frac{(\gamma_1(j) + \ldots + \gamma_M(j))!}{\prod_{i=1}^{M} \gamma_i(j)!} = M^{\aleph}. \tag{4.49}$$

Fortunately, the per-round complexity of DECA given in (4.48) is much smaller that $M^{\aleph}$, i.e. $\Lambda_{max} = o(M^{\aleph})$. In practical product code design, we will also have $\Lambda_{max} \ll M^{\aleph} \ll M^{N^c}$.

The proposed edge coloring algorithm aims at maximizing $\eta(\phi)$ but does not guarantee that $\forall e \in E^c, \rho(e) < \infty$. In some cases, the algorithm may terminate all its rounds with some edges having an infinite order, i.e. the coloring is not double-diversity. This occurs when trying to design a product code with a coding rate very close or equal to $1 - 1/M$, the block-fading/block-erasure Singleton bound rate. To remedy for this weakness, DECA is endowed with an extra subroutine called *Max Diversity*, as shown in Figure 4.6. Likewise the second step in the differential evolution loop, this subroutine applies color permutations to a subset $B_{\aleph_1}$ of edges, $|B_{\aleph_1}| = \aleph_1$, $B_{\aleph_1} \subset B^{\infty}$, and

$$B^{\infty} = \{e \in E^c : \rho(e) = \infty\}. \tag{4.50}$$

4.4.3 Applications

Now, let us apply DECA to design two double-diversity product codes with MDS components. Numerical values are selected to make these codes suitable to distributed storage applications and to diversity systems in wireless networks. The parameter $MaxIter$ is 100. DECA with its hundred iterations runs in a small fraction of a second on a standard computer machine.

Example 4.7. The first application of DECA is to color edges in the compact graph of $C_{P1} = [n, k, d]_q^{\otimes 2}$, where $n = 12$, $k = 10$, $d = 3$, and the finite-field alphabet size is $q > 12$. The coding rate of C_{P1} is $R(C_{P1}) = 25/36 < 1 - 1/M = 3/4$, i.e. the gap to (4.7) is $1/18$. This small gap is enough to render an uncomplicated double-diversity design. The coloring in $\Phi(E^c)$ can be easily converted into its counterpart in $\Phi(E)$ by replacing each supersymbol with 4 symbols. From (4.5) and (4.6), the total number of edge colorings is $|\Phi(E)| \approx 10^{83}$ in the non-compact graph and $|\Phi(E^c)| \approx 10^{19}$ in the compact graph. The differential evolution parameter $\aleph$ is set to 8. The diversity subroutine is deactivated. We have

$$\Lambda_{max}(8, 4) = 2520 \ll |\Phi(E^c)| \ll |\Phi(E)|.$$

For almost any choice of the initial coloring ϕ_0 uniformly distributed in $\Phi(E^c)$, DECA yields a double-diversity coloring ϕ_1. For roughly one choice out of three for ϕ_0, the

G	B	B	B	Y	R
G	B	R	G	Y	G
B	G	G	Y	B	R
G	Y	Y	R	R	B
Y	B	R	Y	G	Y
R	R	R	Y	G	B

(a)

1r	2b	1c	1c	1r	1r
2b	1r	1r	1c	1r	1c
1c	1c	1c	1r	1c	1r
1r	1c	1c	1c	1c	1r
1c	1r	1r	2b	1r	1c
1c	1c	2b	1r	1r	1r

(b)

Figure 4.7: Compact coloring matrix (figure a) and the corresponding rootcheck-order matrix (figure b) for the $[12, 10]^{\otimes 2}$ product code C_{P1} found by DECA, $\eta(\phi) = 32$ and $\rho_{max} = 2$.

algorithm outputs a coloring ϕ_1 such that $\eta(\phi_1) \geq 28$. Figure 4.7(a) shows the matrix representation of a special ϕ_1 found by DECA. It has $\eta(\phi_1) = 32$ which corresponds to $\eta = 128$ in (V_1, V_2, E). The corresponding rootcheck order matrix is shown in Figure 4.7(b). The highest attained order for this coloring is $\rho_{max}(\phi) = 2$. The maximal order for all colorings in $\Phi(E^c)$ from Theorem 4.1 is $\rho_u = 5$. This coloring satisfies equality in (4.8) since $2\rho_{max}(\phi) + \eta_{min}(\phi) = 12$.

Example 4.8. The second more challenging application of DECA is the design of a double-diversity product code attaining the block-fading/block-erasure Singleton bound. Let us consider $C_{P2} = [n_1, k_1, d_1]_q \otimes [n_2, k_2, d_2]_q$, where $n_1 = 14$, $k_1 = 12$, $n_2 = 16$, $k_2 = 14$, $d_1 = d_2 = 3$, and the finite-field alphabet size is $q > 16$. The coding rate is $R(C_{P2}) = 1 - 1/M = 3/4$. From (4.5) and (4.6), the total number of edge colorings is $|\Phi(E)| \approx 10^{131}$ in the non-compact graph and $|\Phi(E^c)| \approx 10^{31}$ in the compact graph. The differential evolution parameter $\aleph$ is set to 7. The diversity subroutine is activated with $\aleph_1 = 8$. We have

$$\Lambda_{max}(7, 4) + \Lambda_{max}(8, 4) = 3150 \ll |\Phi(E^c)| \ll |\Phi(E)|.$$

The initial coloring ϕ_0 is taken to be uniformly distributed in $\Phi(E^c)$. For almost three ϕ_0 choices out of four, DECA yields a double-diversity coloring ϕ_1. Roughly one ϕ_0 choice out of two guarantees $\eta(\phi_1) \geq 34$. Figure 4.8(a) shows the matrix representation of a special ϕ_1 found by DECA. It has $\eta(\phi_1) = 40$ which corresponds to $\eta = 160$ in (V_1, V_2, E). The rootcheck order matrix is shown in Figure 4.8(b). The highest attained order for this coloring is $\rho_{max}(\phi) = 3$. The maximal order for all colorings in $\Phi(E^c)$ from Theorem 4.1 is $\rho_u = 7$. This coloring satisfies $2\rho_{max}(\phi) + \eta_{min}(\phi) = 16$ while the right term in (4.8) is 17.

R	R	R	G	B	R	R	Y
G	R	B	G	R	Y	G	G
B	R	G	B	Y	Y	B	B
R	Y	G	Y	B	Y	Y	Y
R	G	G	G	G	G	B	Y
R	B	B	G	B	B	B	Y
Y	R	Y	R	B	Y	G	R

(a)

2c	3b	1c	1r	1r	1c	1c	1r
1c	2r	1r	3r	1c	1r	2c	1c
1c	1r	1r	1c	1c	2r	2r	1c
1r	1c	1r	1c	1r	3b	1c	2c
1r	1c	2c	3r	1c	1c	1r	1r
1r	1c	2c	1r	2c	1c	3b	1r
1c	2r	1c	1c	1r	2r	1r	1c

(b)

Figure 4.8: Compact coloring matrix (figure a) and the corresponding rootcheck-order matrix (figure b) for the $[14, 12] \otimes [16, 14]$ product code C_{P2} found by DECA, $\eta(\phi) = 40$ and $\rho_{max} = 3$.

Example 4.9. A third example suitable for nowadays distributed storage warehouses is $C_{P3} = [10, 8, 3]_q \otimes [10, 9, 2]_q$. The coding rate is $R = 18/25$ with a minimum distance $d_1 d_2 = 6$ and the locality is $n_1 = n_2 = 10$, i.e. this code is an improvement to the standard $RS[14, 10]$ used by Facebook [80]. The coloring ensembles have sizes $|\Phi(E)| \approx 10^{57}$ and $|\Phi(E^c)| \approx 10^{27}$ respectively. The DECA algorithm produced double-diversity edge colorings where we distinguish two classes: a first class of colorings with $\rho_{max} = 3$ and $\eta(\phi) = 41$, and a second class with $\rho_{max} = 2$ and $\eta(\phi) = 40$. An edge coloring of the second class is shown in Figure 4.9. The reader is invited to determine the rootcheck order matrix and verify that 40 super-edges have root order 1 and 10 super-edges have a root order equal to 2.

G	Y	R	B	Y	B	B	Y	Y	B
R	R	R	Y	G	Y	Y	B	R	R
Y	B	Y	R	R	R	R	R	G	Y
G	G	G	B	G	R	G	G	G	Y
B	B	B	G	B	G	Y	R	B	G

Figure 4.9: Compact coloring matrix for the $[10, 8] \otimes [10, 9]$ product code found by DECA, $\eta(\phi) = 40$ and $\rho_{max} = 2$.

In the figures of previous examples, the four colors were also indicated by the first letter of the color name, Red, Green, Blue, and Yellow. The rootcheck order $\rho(e)$ for an edge e in E^c (which is also the order of the four code symbols associated to that edge) is indicated by an integer in the right part of each figure for the first two examples. In the rootcheck order matrix, $2r$ means that this supersymbol has order 2 and its rootcheck node is a row. Similarly, $2c$ designates a supersymbol with order 2 and a column rootcheck. The letter 'b' is written when a supersymbol has both rootchecks, a row and a column rootcheck.

Product codes in Examples 4.7-4.9 do not satisfy the ρ_u condition given in (4.15) and the sufficient condition of Lemma 4.5 either. An interesting question arises. Does an edge coloring with $\rho_{max} = 1$ exist for a 6×6 compact graph? We provide a partial answer in the sequel. A similar answer is valid for the 7×8 compact graph.

The 6×6 compact graph is perfectly balanced. Let us start with the first color 'R'. The unique solution to get $\rho(e) = 1$ for all edges e with $\phi(e) = R$ is to place 'R' entries separately on the first row and the first column. Hence, no row or a column contain the same color twice. The first 9 edges are located as follows:

$$\begin{bmatrix} & R & R & R & R & R \\ R & & & & & \\ R & & & & & \\ R & & & & & \\ R & & & & & \end{bmatrix}. \tag{4.51}$$

We start over with the second color 'G' using the same rule. Given the lack of space on the second row and the second column, the ninth green edge is placed on the top left corner. We get

$$\begin{bmatrix} G & R & R & R & R & R \\ R & & G & G & G & G \\ R & G & & & & \\ R & G & & & & \\ R & G & & & & \\ & G & & & & \end{bmatrix}. \tag{4.52}$$

At this point, 18 super-edges have a rootcheck order $\rho = 1$. Seven edges only can be colored in blue, three edges on the third row, three edges on the third column, and one edge at the intersection of the second row and the second column. One color 'R' can be moved down to the last row leading to the following coloring:

$$\begin{bmatrix} G & R & R & R & R & B \\ R & B & G & G & G & G \\ R & G & & B & B & B \\ R & G & B & & & \\ R & G & B & & & \\ B & G & B & & & R \end{bmatrix}. \tag{4.53}$$

Finally, we reached an edge coloring where all edges of three colors satisfy $\rho(e) = 1$. Unfortunately, there is no space left for edges of 'Y' to achieve $\rho(e) = 1$. The situation is even worse, the remaining edges for 'Y' make five primitive stopping sets (three 2×2, one 2×3, and one 3×2). This edge coloring has no diversity.

4.4.4 Random edge coloring

The efficiency of the DECA algorithm was validated in the previous section in terms of number of edges of first order and the maximal order over all edges. Clearly, while evolving from one coloring to another in order to get a large $\eta(\phi)$, DECA also produced a very small maximal order $\rho_{max}(\phi)$. Any deterministic construction seems to be destined to fail given the huge size of the ensembles $\Phi(E)$ and $\Phi(E^c)$.

In this sub-section, another way to show the efficiency of our coloring algorithm is to make random selections from $\Phi(E)$ and $\Phi(E^c)$ and get an estimate of the probability distributions of $\eta(\phi)$ and $\rho_{max}(\phi)$. Indeed, a uniformly distributed permutation in the symmetric group of order N yields a uniformly distributed edge coloring ϕ in $\Phi(E)$. This is also true for $\Phi(E^c)$ when the symmetric group has order N^c. Thus, in a uniform manner, we selected 2 billion edge colorings through our computer application from $\Phi(E)$ and $\Phi(E^c)$ respectively. For each coloring, rootcheck orders of all edges were computed, i.e. for the N edges in the non-compact graph and the N^c edges in the compact graph. Only double-diversity colorings are counted in this comparison, i.e. colorings with at least one edge of infinite rootcheck order are excluded. As an illustration, the characteristics of double-diversity random coloring for C_{P1} are plotted in Figure 4.10 where numerical estimations of all probability distributions are compared to colorings designed via DECA.

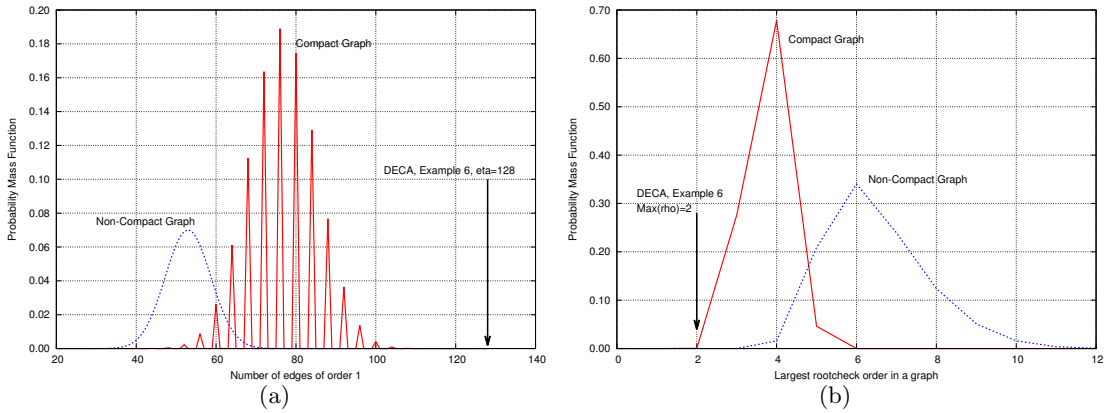


Figure 4.10: Distribution of $\eta(\phi)$ (figure a) and $\rho_{max}(\phi)$ (figure b) for double-diversity random edge colorings uniformly distributed in $\Phi(E)$ and $\Phi(E^c)$. Product code $[12, 10]^{\otimes 2}$.

Double diversity design is more arduous for the rate-3/4 C_{P2} product code than for the rate-25/36 C_{P1} product code because of the rate-diversity tradeoff given by the Singleton bound. For C_{P1}, the $[12, 10]^{\otimes 2}$ code, 8.97% of uniformly sampled colorings have double diversity in $\Phi(E^c)$, whereas this fraction is 43.6% in $\Phi(E)$. For C_{P2}, the $[14, 12] \otimes [16, 14]$ code, only 0.00039% of uniformly sampled colorings have double diversity in $\Phi(E^c)$, and we found no double-diversity colorings in $\Phi(E)$ despite the 2 billion samples. As expected, compact graphs exhibit better characteristics than non-compact graphs thanks to their simpler structure, i.e. $n_i - k_i$ parity symbols are grouped inside a unique supersymbol: for C_{P1}, one double-diversity random coloring has $\eta(\phi) = 88$, $\rho_{max}(\phi) = 4$ for non-compact graphs, seven double-diversity colorings have $\eta(\phi) = 120$, and $\rho_{max}(\phi) = 2$ for compact graphs. There exists a double-diversity coloring in $\Phi(E)$ with $\rho_{max}(\phi) = 3$ but its η is 85. The estimated probability mass functions for C_{P1} are plotted in Figures 4.10(a) and 4.10(b). For C_{P2}, one double-diversity random coloring reached $\eta(\phi) = 128$ and $\rho_{max}(\phi) = 4$ out of the 2 billion samples from $\Phi(E^c)$. In all cases, for both η and ρ, double-diversity random colorings are not as efficient as colorings designed via the DECA algorithm. The situation is worse for random colorings if a double-diversity code with maximal rate $1 - 1/M$ is to be designed. The DECA algorithm exhibits excellent values, $\eta = 160$ and $\rho = 3$, for the rate-3/4 $[14, 12] \otimes [16, 14]$ product code.

4.5 Code performance in presence of erasures

Iterative decoding performance of $C_P = C_1 \otimes C_2$ is studied in presence of channel erasures, with and without edge coloring. The iterative decoder makes row and column iterations where the component decoder of C_i can be an algebraic erasure-filling decoder (limited by $d_i - 1$) or a maximum-likelihood decoder of C_i. As stated in Section 4.3.1, type II and type III stopping sets are identical because the non-binary codes C_1 and C_2 are MDS. The word error probability of the iterative decoder is denoted by $P_{ew}^{\mathcal{G}}$. The product code can also be decoded via an ML decoder, i.e. maximum likelihood decoding of C_P based on a Gaussian reduction of its parity-check matrix. The word error probability under ML decoding of C_P is denoted by P_{ew}^{ML}.

4.5.1 Block erasures

Consider the block-erasure channel $CEC(q, \epsilon)$. The N symbols of a codeword are partitioned into M blocks, each block contains symbols associated to edges in $\mathcal{G}$ with the same color. The $CEC(q, \epsilon)$ channel erases a block with a probability ϵ. The block is correctly received with a probability $1 - \epsilon$. Erasure events are independent from one block to another. We say that a *color is erased* if the associated block of N/M symbols is erased. Assume that $\mathcal{G}$ is endowed with a double-diversity edge coloring ϕ (i.e. $L(\phi) = 2$) as defined in Corollary 4.1. Then, on the block-erasure channel $CEC(q, \epsilon)$, for a rate satisfying

$$1 - \frac{2}{M} < R \leq 1 - \frac{1}{M}, \tag{4.54}$$

we have

$$\epsilon^2 \leq P_{ew}^{ML} \leq P_{ew}^{\mathcal{G}} \leq \sum_{i=2}^{M} \binom{M}{i} \epsilon^i (1-\epsilon)^{M-i}. \tag{4.55}$$

Since ϕ has a double diversity, there exist two colors among the M colors such that the iterative decoder must fail if both colors are erased. This explains the upper bound of $P_{ew}^{\mathcal{G}}$ in (4.55). The upper bound is valid for any rate less than the maximal achievable rate for double diversity, i.e. $1 - \frac{1}{M}$. Now, since $R > 1 - \frac{2}{M}$, the ML decoder for C_P cannot attain a diversity $L = 3$ otherwise the block-fading/block-erasure Singleton bound would be violated. Consequently, the ML decoder of C_P can only reach $L = 2$ and so there exists a pair of erased colors that cannot be solved by the ML decoder. This explains the lower bound in (4.55). The reader can easily verify that

$$\lim_{\epsilon \to 0} \frac{\log P_{ew}^{ML}}{\log \epsilon} = \lim_{\epsilon \to 0} \frac{\log P_{ew}^{\mathcal{G}}}{\log \epsilon} = L = 2. \tag{4.56}$$

The slope of P_{ew} versus the erasure probability ϵ in a double-logarithmic scale is equal to 2. Under the stated constraint on R, the upper bound in (4.55) is the exact expression of the outage probability on a block-erasure channel valid for q-ary codes with asymptotic length [41]. For double-diversity edge colorings found by DECA in Examples 4.7 and 4.8, $P_{ew}^{\mathcal{G}}$ equals its upper bound in (4.55). These examples achieve the outage probability although a code may perform better than the outage probability at finite length. For these colorings where $M = 4$, the error probability on $CEC(q, \epsilon)$ behaves like $P_{ew}^{\mathcal{G}} = 6\epsilon^2 + O(\epsilon^3)$. One possible interpretation of this behavior is: the optimization of $\eta(\phi)$ (equivalent in some sense to minimizing $\rho(\phi)$) pushed the performance of edge colorings found by the DECA algorithm as far as possible from the lower bound ϵ^2. As can be observed in Figures 4.7 and 4.8, all rows and all columns include the four colors. When any two colors out of four are erased, the iterative decoder will completely fail without correcting a single supersymbol. A double-diversity edge coloring guarantees that all stopping sets are covered by at least two colors but it cannot cover all stopping sets with three colors or more otherwise we get $L = 3$ which contradicts $R > 1 - \frac{2}{M}$. Fortunately, these product codes are diversity-wise MDS and the second code in Example 4.8 has the maximal coding rate for double diversity. In the sequel, we will see that these codes also perform well in presence of independent erasures.

4.5.2 Independent erasures

Consider the i.i.d. erasure channel $SEC(q, \epsilon)$. The N symbols of a codeword are independently erased by the channel. A symbol is erased with a probability ϵ and is correctly received with a probability $1 - \epsilon$. Edge coloring has no effect on the performance of C_P on the $SEC(q, \epsilon)$ channel. Before studying the performance on the $SEC(q, \epsilon)$, following Examples 4.3 & 4.4 and Theorems 4.2 & 4.3, we state an obvious result about obvious stopping sets in the following proposition.

Proposition 4.3. *Let $C_P = C_1 \otimes C_2$ be a product code with non-binary MDS components. All obvious stopping sets are supports of product code codewords.*

Proof. Consider an $\ell_1 \times \ell_2$ obvious stopping set. Its rectangular support is $\mathcal{R}(\mathcal{S}) = \mathcal{R}_1(\mathcal{S}) \times \mathcal{R}_2(\mathcal{S})$. We have $\ell_1 \geq d_1$ and $\ell_2 \geq d_2$. From Proposition 4.2, there exists a column codeword $x = (x_1, x_2, \ldots, x_{n_1}) \in C_1$ of weight ℓ_1 with support $\mathcal{R}_1(\mathcal{S}) \times \{j_1\}$, where $j_1 \in \mathcal{R}_2(\mathcal{S})$. Similarly, there exists a row codeword $y = (y_1, y_2, \ldots, y_{n_2}) \in C_2$ of weight ℓ_2 with support $\{i_1\} \times \mathcal{R}_2(\mathcal{S})$, where $i_1 \in \mathcal{R}_1(\mathcal{S})$. Now, the Kronecker product of x and y satisfies $\mathcal{X}(x \otimes y) = \mathcal{S}$. □

Corollary 4.5. *Consider a product code $C_P = C_1 \otimes C_2$ with non-binary MDS component codes. Assume the symbols of C_P are transmitted over a $SEC(q, \epsilon)$ channel. Then, for $\epsilon \ll 1$, the error probabilities satisfy $P_{ew}^{\mathcal{G}} \sim P_{ew}^{ML}$.*

Proof. On the $SEC(q, \epsilon)$, the word error probabilities are given by [93],

$$P_{ew}^{ML} = \sum_{i=d_1 d_2}^{N} \Psi_i(ML) \epsilon^i (1-\epsilon)^{N-i}, \tag{4.57}$$

where $\Psi_i(ML)$ is the number of weight-i erasure patterns covering a product code codeword, and

$$P_{ew}^{\mathcal{G}} = \sum_{i=d_1 d_2}^{N} \Psi_i(\mathcal{G}) \epsilon^i (1-\epsilon)^{N-i}, \tag{4.58}$$

where $\Psi_i(\mathcal{G})$ is the number of weight-i erasure patterns covering a stopping set. Of course, here we refer to stopping sets in the non-compact graph $\mathcal{G}$, i.e. in the $n_1 \times n_2$ product code matrix. Next, since N is fixed (asymptotic length analysis is not considered in this chapter) we write $P_{ew}^{ML} = \Psi_{d_1 d_2}(ML)\epsilon^{d_1 d_2} + o(\epsilon^{d_1 d_2})$ and $P_{ew}^{\mathcal{G}} = \Psi_{d_1 d_2}(\mathcal{G})\epsilon^{d_1 d_2} + o(\epsilon^{d_1 d_2})$. From Proposition 4.3, we get the equality $\Psi_{d_1 d_2}(\mathcal{G}) = \Psi_{d_1 d_2}(ML)$ and so we obtain $\lim_{\epsilon \to 0} P_{ew}^{\mathcal{G}} / P_{ew}^{ML} = 1$. □

The erasure patterns can be decomposed according to the size of the covered stopping set. The coefficient $\Psi_i(\mathcal{G})$ becomes $\Psi_i(\mathcal{G}) = \sum_{w=d_1 d_2}^{i} \Psi_{i,w}(\mathcal{G})$, where $\Psi_{i,w}(\mathcal{G})$ is the number of weight-i patterns covering a stopping set of size w. It is clear that $\Psi_{w,w}(\mathcal{G}) = \tau_w$. For small $i - w$, $\Psi_{i,w}(\mathcal{G})$ can be approximated by $\sum_{\mathcal{A}} \binom{N-\mathcal{A}}{i-w} \tau_{w,\mathcal{A}}$, where $\tau_{w,\mathcal{A}}$ is the number of stopping sets of size w having $|\mathcal{R}(\mathcal{S})| = \mathcal{A}$. For $w \leq d_1 d_2 + d_1 + d_2 + 1$, the area $\mathcal{A}$ is bounded from above by the product $\ell_1^0 \times \ell_2^0$ from Lemma 4.1. Numerical evaluations of $\Psi_i(\mathcal{G})$ are tractable for very short codes ($N \leq 25$) and become very difficult for codes of moderate size and beyond, e.g. $N = 144$ and $N = 224$ for the $[12, 10]^{\otimes 2}$ and the $[14, 12] \otimes [16, 14]$ codes respectively. For this reason, expressions (4.57) and (4.58) are not practical to predict the $SEC(q, \epsilon)$ performance of product codes with significant characteristics.

For $P_{ew}^{\mathcal{G}}$, thanks to Theorems 4.2 and 4.3, a union bound can be easily established. Indeed, we have

$$P_{ew}^{\mathcal{G}} = Prob(\exists \mathcal{S} \; covered) \leq \sum_{w} Prob(\exists \mathcal{S} : |\mathcal{S}| = w, \mathcal{S} \; covered),$$

leading to

$$P_{ew}^{\mathcal{G}} \leq P^U(\epsilon) = \sum_{w=d_1 d_2}^{N} \tau_w \epsilon^w. \tag{4.59}$$

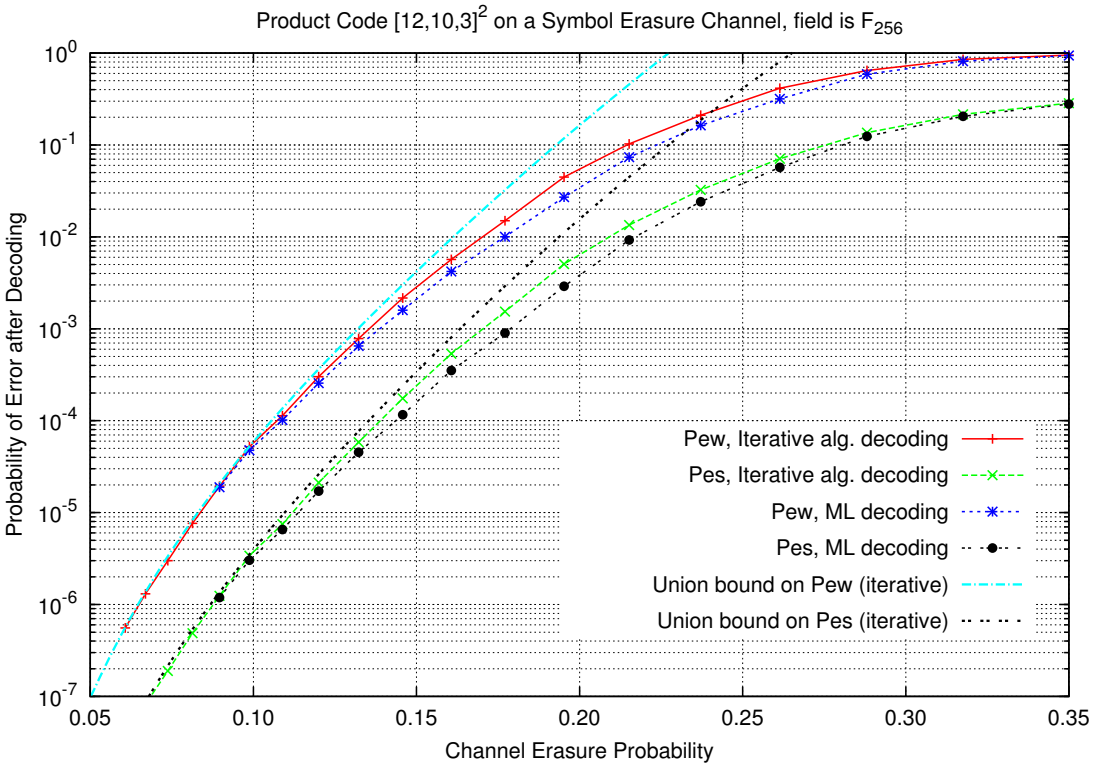


Figure 4.11: Product code $[12, 10]_q^{\otimes 2}$, no edge coloring. Word and symbol error rate performance for iterative decoding versus its union bound and ML decoding.

From Theorem 4.2, the union bound $P^U(\epsilon)$ for the $[12, 10, 3]_q^{\otimes 2}$ product code is

$$\begin{aligned} P^U(\epsilon) =& 48400\epsilon^9 + 6098400\epsilon^{12} + 23522400\epsilon^{13} + 17641800\epsilon^{14} \\ &+ 1754335440\epsilon^{15} + 9126691200\epsilon^{16} + o(\epsilon^{16}). \end{aligned}$$

The performance of this code on the $SEC(q, \epsilon)$ channel is shown in Figure 4.11. We used the standard finite field of size $q = 256$. The union bound for the symbol error probability $P_{es}^{\mathcal{G}}$ is derived by weighting the summation term in (4.59) with w/N, i.e. $P_{es}^{\mathcal{G}} \leq \sum_{w=d_1 d_2}^{N} \frac{w}{N} \tau_w \epsilon^w$. As observed in the plot of Figure 4.11, the union bound is sufficiently tight. Furthermore, the performance of the iterative algebraic row-column decoder is very close to that of ML decoding in the whole range of ϵ. For small ϵ, the curves are superimposed as predicted by Corollary 4.5.

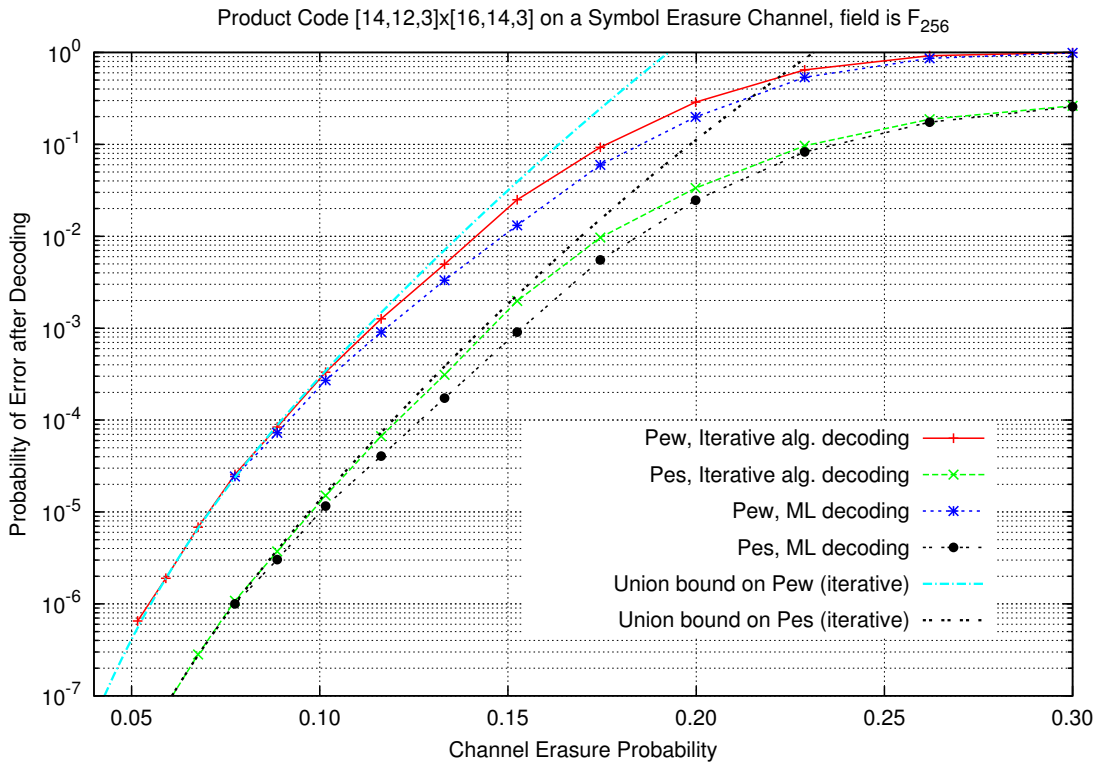


Figure 4.12: Product code $[14, 12]_q \otimes [16, 14]_q$, no edge coloring. Word and symbol error rate performance for iterative decoding versus its union bound and ML decoding.

The union bound $P^U(\epsilon)$ for the $[14, 12, 3]_q \otimes [16, 14, 3]_q$ product code is

$$P^U(\epsilon) = 203840\epsilon^9 + 44946720\epsilon^{12} + 174894720\epsilon^{13} + 131171040\epsilon^{14} + 17839261440\epsilon^{15} + 126887941180\epsilon^{16} + o(\epsilon^{16}).$$

The performance of this code on the $SEC(q, \epsilon)$ channel is shown in Figure 4.12. Similar to the previous code, the union bound is tight enough and iterative decoding performs very close to ML decoding. Finally, let us interpret these results from a finite-length information theoretical point of view [74]. The $SEC(q, \epsilon)$ of Shannon capacity $\log_2(q)(1-\epsilon)$ behaves exactly like a $BEC(\epsilon)$ of capacity $(1 - \epsilon)$ but erasures in the SEC occur at the symbol level instead of the binary digit level. Finite-regime BEC bounds from [74] are directly applicable to our product codes over the $SEC(q, \epsilon)$. The BEC channel dispersion is $V = \epsilon(1 - \epsilon)$ and its maximal achievable rate is given by [74], Theorem 53,

$$R = (1 - \epsilon) - \sqrt{\frac{V}{n}} Q^{-1}(P_{ew}) + O(\frac{1}{n}), \tag{4.60}$$

where n is the code length, $Q(x)$ is the Gaussian tail function, ϵ is the channel erasure

probability, and P_{ew} is the target word error probability. The next table shows how good is the proposed product code based on MDS components.

	Coding Rate R for $\epsilon = 0.15$	Erasure Prob. ϵ for $R = 0.75$
Polyanskiy-Poor-Verdú	$0.794 : P_{ew} = 1.0 \cdot 10^{-2}$	0.189
$[14, 12]_q \otimes [16, 14]_q$	$0.750 : P_{ew} = 1.0 \cdot 10^{-2}$	0.150
Regular-$(3, 12)$ LDPC	$0.750 : P_{ew} = 2.9 \cdot 10^{-2}$	0.135

Table 4.3: Finite-length performance of the $[14, 12]_q \otimes [16, 14]_q$ product code. The value of ϵ in the third column is given for $P_{ew} = 10^{-2}$ at all rows.

4.5.3 Unequal probability erasures

In communication and storage systems, erasure events of unequal probabilities may occur. In order to observe the effect of a double-diversity coloring on the performance in multiple erasure channels, we define the $SEC(q, \{\epsilon_i\}_{i=1}^M)$. On this channel, symbol erasure events are independent but the probability of erasing a symbol is ϵ_i if it is associated to an edge in $\mathcal{G}$ with color $\phi(e) = i$. The union bound is easily modified to get

$$P_{ew}^{\mathcal{G}} \leq P^U(\epsilon_1, \ldots, \epsilon_M), \tag{4.61}$$

where

$$P^U(\epsilon_1, \ldots, \epsilon_M) = \sum_{w=d_1 d_2}^{N} \sum_{\substack{w_1, \ldots, w_M \\ : \sum_i w_i = w}} \tau(w_1, \ldots, w_M) \prod_{i=1}^{M} \epsilon_i^{w_i}. \tag{4.62}$$

The coefficient $\tau(w_1, \ldots, w_M)$ is the number of stopping sets of size $w = \sum_{i=1}^M w_i$, where i symbol edges have color i, $i = 1 \ldots M$. Clearly, the coefficients $\tau(w_1, \ldots, w_M)$ depend on the edge coloring ϕ. For double-diversity colorings and $M \geq 2$, these coefficients satisfy the following property:
For any stopping set $\mathcal{S}$ such that $|\mathcal{S}| = w$, $\tau(w_1, \ldots, w_M)$ does exist for $\sum_{i=1}^M w_i = w$ and $w_i > 0$ only, i.e. no weak compositions of w are authorized by ϕ.
Hence, the product code should perform well if one of the ϵ_i is close to 1 and the remaining ϵ_i are small enough. The extreme case is true thanks to double diversity yielding $P^U(0^{M-1}, 1^1) = 0$, where $(0^{M-1}, 1^1)$ represents all vectors with all positions at 0 except for one position set to 1. Figure 4.13 shows the performance of $[12, 10]_q^{\otimes 2}$ on the $SEC(q, \{\epsilon_i\}_{i=1}^M)$ channel with $M = 4$ colors. The edge coloring is the double-diversity coloring produced by the DECA algorithm and drawn in Figure 4.7. The expression of $P^U(\epsilon_1, \ldots, \epsilon_M)$ is determined by stopping sets enumeration as in Theorems 4.2 and 4.3. Details are omitted and the very long expression of $P^U(\epsilon_1, \ldots, \epsilon_M)$ is not shown. The special case $\epsilon_1 = \epsilon_2 = \epsilon_3$ is considered and the performance is plotted as a function of

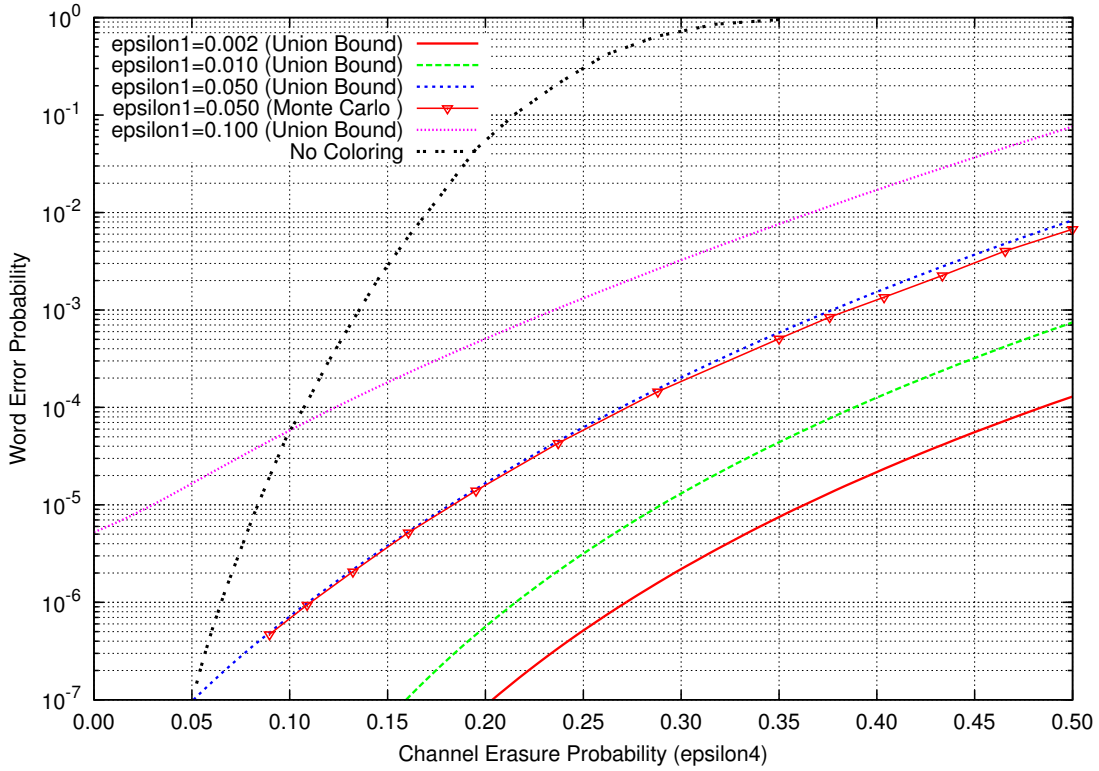


Figure 4.13: Product code $[12, 10]_q^{\otimes 2}$ with double-diversity edge coloring. Word error rate performance versus ϵ_4, for iterative decoding on the $SEC(q, \{\epsilon_i\}_{i=1}^4)$ channel with $\epsilon_1 = \epsilon_2 = \epsilon_3$.

ϵ_4. For a fixed ϵ_1, double diversity dramatically improves the performance with respect to ϵ_4.

4.6 Conclusions

Non-binary product codes with MDS components are studied in this chapter in the context of iterative row-column algebraic decoding. Channels with both independent and block erasures are considered. The rootcheck concept and associated double-diversity edge colorings were described after introducing a compact graph representation for product codes. For solving erased symbols, an upper bound of the number of decoding iterations is given as a function of the graph size and the color palette size M. Stopping sets are defined in the context of MDS components and a relationship is established with the graph representation of the product code. A full characterization of these stopping sets is given up to a weight $(d_1 + 1)(d_2 + 1)$. Then, we proposed a differential evolution edge coloring algorithm to design colorings with a large population of minimal rootcheck

order symbols. The complexity of this algorithm per iteration is $o(M^{\aleph})$, where $\aleph$ is the differential evolution parameter. The performance of MDS-based product codes with and without double-diversity coloring is analyzed. In addition, ML and iterative decoding are proven to coincide at small channel erasure probability.

A complete enumeration of product code codewords is still an open problem in coding theory. Following the enumeration of bipartite graphs in Section 4.3.4 (see also Table 4.1) and following the DECA algorithm that aims at improving $\eta(\phi)$ in Section 4.4.2, two open problems can be stated.

• In number theory. There exists no recursive or closed form expression for the special partition function, i.e. the number of special partitions of an integer. Also, in a way similar to the Hardy-Ramanujan formula, the asymptotic behavior is unknown for the number of special partitions. Special partitions are introduced in Definition 4.10.

• In graph theory and combinatorics. Consider a matrix of size $H \times W$ and a coloring palette of size M. For simplicity, assume that $H \cdot W$ is multiple of M. A matrix entry is called *edge*. A color is assigned to each edge in the matrix. All M colors are equally used. A matrix edge/entry (i, j) of color c is said to be *good* if it is the unique entry with color c either on row i or on column j. The number of good entries is denoted by $\eta(\phi)$, see also (4.43). Given the matrix height H, width W, and the palette size M, find the maximum achievable number of good entries $\eta(\phi)$ over the set of all edge colorings ϕ. A simpler problem would be to find an upper bound of $\eta(\phi)$.

Conclusions and perspectives

Conclusions

This Ph.D thesis investigates distributed storage over data centers interacting with cloud computing resources. In distributed storage the erasure model is combined to special coding structures. We extensively used the erasure channel model in our research work. The erasure channel is simple, suited to both finite-length and infinite-length analysis of channel coding. Despite its simplicity, it brings a great comprehension of the code behavior and helps in designing good codes. We used the erasure model in spatial coupling and in coding for diversity.

The first part of this work is devoted to performance study of spatially coupled codes for erasure channel. Spatial coupling of codes on graphs, mainly LDPC codes, is a recent method to attain channel capacity for any memoryless symmetric binary-input channel. Shannon capacity is approached by letting three parameters grow large enough: The graph degrees, the coupling window, and the chain length. In our work, we showed that a small coupling window yields impressive results in terms of iterative decoding thresholds. This new method is called forward layered coupling. The method is inspired from overlapped layered coding and has the shortest possible memory, i.e. $w = 2$. Edges of local ensembles and those defining the spatial coupling are separately built. The new method also allows the construction of non-uniform coupling chains with near-Shannon spatially-varying thresholds under iterative decoding by gluing non-equal LDPC protographs which leads to a spatially-variable BP threshold.
Then we introduced a novel forward-layered coupling schemes for Root-LDPC. Local protographs are coupled by new edges connected to their parity bits only. Spatial coupling of parity bits plays the role of doping for full-diversity codes. The erasure channel performance indicate an enhancement of the code outage boundary with respect to standard uncoupled codes. The spatially-coupled Root-LDPC ensembles achieve threshold boundaries very close to the capacity boundary (saturation) in the erasure plane. This behavior in the erasure plane will automatically result in a saturation of the root-LDPC outage boundary on a real block-fading channel with additive noise.

Spatially-coupled Root-LDPC are applied to storage and diversity applications. From Definition 2.1 and the structure of the double-diversity Root-LDPC codes, it is obvious that information bits have locality $d_c - 1$, i.e. the degree of check nodes minus one edge, in the coupling scheme presented in Section 3.4.3 In the special case where erasures occur per color (block erasures) the locality of a random LDPC ensemble is $O(\log(n))$ whereas that of a Root-LDPC is maintained at $d_c - 1$ thanks to rootcheck nodes of order 1. Usually, we would like to maintain all information bits at root order 1. This forces the construction to couple parity bits only. We also observed that the outage boundary is not completely saturated towards the capacity line. Root order 1 for all information bits is too constraining for rate 3/4 and 4 colors. A potential solution to saturate the boundary of coupled Root-LDPC (as a perspective for future work) is to introduce a fraction of order 2 information bits. This should allow the enhancement of spatial coupling by engaging both information and parity bits.

In the second part of this Ph.D thesis, non-binary product codes with MDS components are studied in the context of iterative row-column algebraic decoding. Channels with both independent and block erasures are considered. The rootcheck concept and associated double-diversity edge colorings were described after introducing a compact graph representation for product codes. The graph representation of product codes, as known since Tanner's pioneering work, is shown to be strong enough in the MDS case. New results about iterative decoding of product codes are obtained. For solving erased symbols, an upper bound of the number of decoding iterations is given as a function of the graph size and the color palette size M. Stopping sets are defined in the context of MDS components and a relationship is established with the graph representation of the product code. A full characterization of these stopping sets is given up to a weight $(d_1 + 1)(d_2 + 1)$. The newly introduced compact graphs for product codes allow us to propose an edge coloring algorithm for double diversity. We proposed a differential evolution edge coloring algorithm to design colorings with a large population of minimal rootcheck order symbols. The complexity of this algorithm per iteration is $o(M^{\aleph})$, where $\aleph$ is the differential evolution parameter. The performance of MDS-based product codes with and without double-diversity coloring is analyzed. In addition, for small channel erasure probability, the iterative decoder probability of error is asymptotic to the ML decoder probability of error. This is a nice property of MDS-based product codes.

Perspectives

As a future work, we propose the following research directions:

- Complete investigations on the relationship between erasure channel and fading channel. The erasure plane representation and the spatial coupling shown in this report are limited to binary codes.
- Starting from the stopping sets enumeration, find tight bounds to the number of product code codewords of a given Hamming weight.
- Interesting open problems were found in Chapter 4:

  $\rightarrow$ In number theory. There exists no recursive or closed form expression for the special partition function, i.e. the number of special partitions of an integer. Also, in a way similar to the Hardy-Ramanujan formula, the asymptotic behavior is unknown for the number of special partitions. Special partitions are introduced in Definition 4.10.
  $\rightarrow$ In graph theory and combinatorics. Consider a matrix of size $H \times W$ and a coloring palette of size M. For simplicity, assume that $H \cdot W$ is multiple of M. A matrix entry is called *edge*. A color is assigned to each edge in the matrix. All M colors are equally used. A matrix edge/entry (i, j) of color c is said to be *good* if it is the unique entry with color c either on row i or on column j. The number of good entries is denoted by $\eta(\phi)$, see also (4.43). Given the matrix height H, width W, and the palette size M, find the maximum achievable number of good entries $\eta(\phi)$ over the set of all edge colorings ϕ. A simpler problem would be to find an upper bound of $\eta(\phi)$.
- Security is not considered in this Ph.D. work. In Root-LDPC literature, a trade-off between security and diversity was established. It would be interesting to redo product codes design for distributed storage taking security into account.

Appendices

4.A Proof of Theorem 4.3

Of course, C_1 and C_2 are interchangeable which explains why we stated the theorem for $d_1 < d_2$. From Lemma 4.1, the maximal rectangle height satisfies $\ell_1^0 \leq (d_1+2)$. Similarly, under the condition $d_2 < 3d_1 - 1$, the maximal rectangle width satisfies $\ell_2^0 \leq (d_2 + 3)$. From $d_1 \times d_2$ up to the maximal size $(d_1+2)\times(d_2+3)$, there are twelve different sizes listed in Figure 4.14. The most right column tells us when sizes located on the same row are equal. Also, the first entries on rows 4 and 5 are equal if $d_2 = d_1+1$. For these rectangular supports, the stopping set weight w takes values from rows 1-4 (and the ranges between these values) in the table drawn in Figure 4.14, i.e. $d_1 d_2 \leq w \leq (d_1 + 1)(d_2 + 1)$.

The proof shall consider $d_2 > 2d_1$ in its sub-section C. There exists no integer d_2 in the range $]2d_1, 3d_1 - 1[$ for $d_1 = 2$. Only for sub-section C and $d_1 = 2$, we consider a rectangular support with a width up to $d_2 + 4$ which enlarges the range of d_2 to $2d_1 < d_2 < 4d_1 - 1$ and permits to keep the case $d_1 = 2$ valid in sub-section C.

- The case $w < d_1 d_2$.
  The proof is similar to $w < d^2$ in Theorem 4.2. Here, we just deduce that $w \geq d_1 \ell_2$ and $w \geq d_1 \ell_2$ leading to the contradiction $w \geq d_1 d_2$. Therefore $\tau_w = 0$ for $w < d_1 d_2$ under type II iterative decoding.

- The case $w = d_1 d_2$.
  We use similar inequalities as in the previous case which resembles the proof in Theorem 4.2 for $w = d^2$. We get that $\mathcal{R}(\mathcal{S}) = \mathcal{S}$. All stopping set of size $d_1 d_2$ are obvious. Their number is given by choosing d_1 rows out of n_1 and d_2 columns out of n_2.

- The case $d_1 d_2 < w < d_1(d_2 + 1)$.
  Given that $\ell_1 \ell_2 \geq w > d_1 d_2$, we get $\ell_1 \geq d_1$ and $\ell_2 \geq d_2$, since $d_1 \times d_2$ is the smallest $\mathcal{R}(\mathcal{S})$. Take $\ell_1 = d_1$, then $\ell_2 \geq d_2 + 1$ because $w > d_1 d_2$. The weight

d_1d_2			
$d_1(d_2+1)$			
$(d_1+1)d_2$	$d_1(d_2+2)$		$d_2 = 2d_1$
$(d_1+1)(d_2+1)$		$d_1(d_2+3)$	$d_2 = 2d_1 - 1$
$(d_1+2)d_2$	$(d_1+1)(d_2+2)$		$d_2 = 2d_1 + 2$
$(d_1+2)(d_2+1)$	$(d_1+1)(d_2+3)$		$d_2 = 2d_1 + 1$
$(d_1+2)(d_2+2)$			
$(d_1+2)(d_2+3)$			

Figure 4.14: Size of the rectangular support $\mathcal{R}(\mathcal{S})$ given in the three left columns. The twelve different sizes are listed in increasing order within each column. The right column of this table indicates when sizes on the same row are equal.

of each column must be at least d_1 giving us $w \geq d_1\ell_2 \geq d_1(d_2+1)$, which is a contradiction unless $\tau_w = 0$. The same arguments hold for $\ell_1 > d_1$.

- The case $w = d_1(d_2+1)$.
  The admissible rectangular support can have all sizes $\ell_1 \times \ell_2$ listed in Figure 4.14 starting from $d_1 \times (d_2+1)$.

  – The smallest $\mathcal{R}(\mathcal{S})$ is $d_1 \times (d_2+1)$. All corresponding stopping sets are obvious. Their number is
  $$\binom{n_1}{d_1}\binom{n_2}{d_2+1}.$$

  – The next $\mathcal{R}(\mathcal{S})$ has size $(d_1+1) \times d_2$. The number of zeros is $\beta = (d_1+1)d_2 - d_1(d_2+1) = d_2 - d_1 > 0$. This result contradicts Lemma 4.2 where $\beta = 0$. Hence, this size of the rectangular support yields no stopping sets, $\tau_w = 0$ in this sub-case.

  – The next $\mathcal{R}(\mathcal{S})$ has size $d_1 \times (d_2+2)$. Again, Lemma 4.2 on the existence of a stopping set tells us that $\beta = 0$, but $\beta = d_1(d_2+2) - d_1(d_2+1) = d_1 > 0$. Then $\tau_w = 0$.

– All rectangle sizes from rows $4-8$ in the table in Figure 4.14 are larger than the previous case and make a contradiction on β unless $\tau_w = 0$.

Starting from this point, d_2 should be compared to $2d_1$ in order to sort the values of the stopping set size as given in the table in Figure 4.14.

4.A.1 Minimum distances satisfying $d_2 < 2d_1$

- The case $d_1(d_2+1) < w < (d_1+1)d_2$.
  The smallest $\mathcal{R}(\mathcal{S})$ has size $(d_1+1)\times d_2$ and the largest has size $(d_1+2)\times(d_2+3)$. All these stopping sets contradict Lemma 4.2 if β is computed from the size of $\mathcal{R}(\mathcal{S})$ and w. Then $\tau_w = 0$.

- The case $w = (d_1+1)d_2$.

  – $\mathcal{R}(\mathcal{S})$ has size $(d_1+1)\times d_2$. Stopping sets are obvious and their number is
  $$\binom{n_1}{d_1+1}\binom{n_2}{d_2}.$$

  – $\mathcal{R}(\mathcal{S})$ has size $d_1\times(d_2+2)$. Lemma 4.2 gives $\beta = 0$ but $\beta = d_1(d_2+2) - (d_1+1)d_2 = 2d_1 - d_2 \geq 1$. Then $\tau_w = 0$ in this sub-case.

  – $\mathcal{R}(\mathcal{S})$ has size $(d_1+1)\times(d_2+1)$. We have $\beta = d_1+1$ and d_2-d_1 columns have no 0. The β zeros should be in a $(d_1+1)\times(d_1+1)$ permutation matrix in the remaining β columns. These stopping sets are not obvious and their number is
  $$(d_1+1)!\binom{d_2+1}{d_2-d_1}\binom{n_1}{d_1+1}\binom{n_2}{d_2+1}.$$

  – All other $\mathcal{R}(\mathcal{S})$ greater than the previous case have $\tau_w = 0$ because of a contradiction on β.

- The case $(d_1+1)d_2 < w < d_1(d_2+2)$.
  Let us write $w = (d_1+1)d_2+\lambda$, where λ belongs to $[1, 2d_1-d_2-1]$. If $d_2 = 2d_1-1$ this range for λ is empty and we obtain $\tau_w = 0$. Then, we consider $d_2 < 2d_1-1$.

  – $\mathcal{R}(\mathcal{S})$ has size $d_1\times(d_2+2)$. The number of zeros is $\beta = d_1(d_2+2)-w > 0$ which contradicts Lemma 4.2. There are no stopping sets for this rectangular size.

  – $\mathcal{R}(\mathcal{S})$ has size $(d_1+1)\times(d_2+1)$. We have $\beta = d_1+1-\lambda \in [d_2-d_1+2, d_1]$. The non-obvious stopping sets are built by selecting β columns and β rows and then embedding any 0-permutation matrix, their number is
  $$(d_1+1-\lambda)!\binom{d_1+1}{\lambda}\binom{d_2+1}{d_1+1-\lambda}\binom{n_1}{d_1+1}\binom{n_2}{d_2+1}.$$

  - All other $\mathcal{R}(\mathcal{S})$ greater than the previous case have $\tau_w = 0$ because of a contradiction on β.

- The case $w = d_1(d_2+2)$.
  - For $\mathcal{R}(\mathcal{S})$ with size $d_1(d_2+2)$, we the following number of obvious stopping sets
$$\binom{n_1}{d_1}\binom{n_2}{d_2+2}.$$
  - The next size for $\mathcal{R}(\mathcal{S})$ to be considered is $(d_1+1)\times(d_2+1)$. The number of zeros is $\beta = d_2 - d_1 + 1$. As usual, these non-obvious stopping sets are constructed by a 0-permutation matrix of size β inside $\mathcal{R}(\mathcal{S})$. Their number is
$$(d_2-d_1+1)!\binom{d_1+1}{d_2-d_1+1}\binom{d_2+1}{d_2-d_1+1}\binom{n_1}{d_1+1}\binom{n_2}{d_2+1}.$$
  - Both $d_1\times(d_2+3)$ and $(d_1+2)\times d_2$ lead to a contradiction on β. Now we consider the rectangle of size $(d_1+1)\times(d_2+2)$. The number of zeros is $\beta = d_2+2$. All columns must have a single 0. Regarding the rows, let r_0, r_1, and r_2 be the number of rows with 0, 1, and 2 zeros respectively. We have $r_0+r_1+r_2 = d_1+1$ and $\beta = 2r_2+r_1$. Combining the two previous equalities yields $2r_0+r_1 = 2d_1-d_2$. Many similar cases where encountered in the proof of Theorem 4.2. The number of these non-obvious stopping sets becomes
$$\sum_{2r_0+r_1=2d_1-d_2}\binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1}\frac{(d_2+2)!}{2^{r_0+d_2-d_1+1}}\binom{n_1}{d_1+1}\binom{n_2}{d_2+2}.$$

- The case $d_1(d_2+2) < w < (d_1+1)(d_2+1)$.
  Let us write $w = d_1(d_2+2)+\lambda$, where λ belongs to the interval $[1, d_2-d_1]$.
  - The smallest rectangle has size $(d_1+1)(d_2+1)$. The number of zeros is $\beta = d_2-d_1+1-\lambda \in [1, d_2-d_1]$. The number of these non-obvious stopping sets is found by counting all $\beta\times\beta$ permutation matrices in all positions,
$$(d_2-d_1+1-\lambda)!\binom{d_1+1}{2d_1-d_2+\lambda}\binom{d_2+1}{d_1+\lambda}\binom{n_1}{d_1+1}\binom{n_2}{d_2+1}.$$
  - The next $\mathcal{R}(\mathcal{S})$ has size $(d_1+1)(d_2+2)$ according to the table in Figure 4.14, since both sizes $d_1(d_2+3)$ and $(d_1+2)d_2$ lead to a contradiction on β. The number of zeros for this rectangular support is $\beta = d_2+2-\lambda \in [d_1+2, d_2+1]$. The (d_2+2) columns satisfy: λ columns have no zero and β columns have a unique zero. As usual, we solve $\beta = 2r_2+r_1$ and $r_0+r_1+r_2 = d_1+1$ to get $2r_0+r_1 = 2d_1-d_2+\lambda$ and $r_2 = r_0+\lambda-d_1-1$. The number of non-obvious stopping sets in this case is
$$\sum_{2r_0+r_1=2d_1-d_2+\lambda}\binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1}\frac{(d_2+2)!}{2^{r_2}\lambda!}\binom{n_1}{d_1+1}\binom{n_2}{d_2+2}.$$

– Rectangular supports larger than $(d_1+1)(d_2+2)$ do not correspond to stopping sets for the given range of w, i.e. $\tau_w = 0$.

- The last case $w = (d_1+1)(d_2+1) \leq d_1(d_2+3)$.

  – The number of obvious stopping sets for the smallest $\mathcal{R}(\mathcal{S})$ is

  $$\binom{n_1}{d_1+1}\binom{n_2}{d_2+1}.$$

  If $d_2 = 2d_1 - 1$ then $(d_1+1)(d_2+1) = d_1(d_2+3)$ and corresponds to the following obvious stopping sets

  $$\binom{n_1}{d_1}\binom{n_2}{d_2+3}.$$

  Similarly, if $d_2 = d_1 + 1$ then $(d_1+1)(d_2+1) = (d_1+2)d_2$ and corresponds to the obvious stopping sets with number

  $$\binom{n_1}{d_1+2}\binom{n_2}{d_2}.$$

  Notice that $d_1 \times (d_2+3)$ and $(d_1+2) \times d_2$ have no non-obvious stopping sets (from Lemma 4.2).

  – The next $\mathcal{R}(\mathcal{S})$ is $(d_1+1)(d_2+2)$. The corresponding number of zeros is $\beta = d_1 + 1$. Similar cases were encountered before. The number of these non-obvious stopping sets is

  $$\sum_{2r_0+r_1=d_1+1} \binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1}\frac{(d_2+2)!}{2^{r_0}(d_2-d_1+1)!}\binom{n_1}{d_1+1}\binom{n_2}{d_2+2}.$$

  – Consider $\mathcal{R}(\mathcal{S})$ with size $(d_1+2)(d_2+1)$. We have $\beta = d_2+1$. If $d_2 > d_1+1$, then we find $\tau_w = 0$ by contradicting arguments on β. But if $d_2 = d_1+1$, the number of non-obvious stopping sets becomes

  $$\sum_{2r_0+r_1=d_2+1} \binom{d_2+1}{r_0}\binom{d_2+1-r_0}{r_1}\frac{(d_1+2)!}{2^{r_0}}\binom{n_1}{d_1+2}\binom{n_2}{d_2+1}.$$

  – Consider $\mathcal{R}(\mathcal{S})$ with size $(d_1+1)(d_2+3)$. We have $\beta = 2(d_1+1)$. If $d_2 < 2d_1-1$ there are no stopping sets. When $d_2 = 2d_1-1$, we get $\beta = 2(d_1+1) = d_2+3$. The number of non-obvious stopping sets is found to be (method as in previous cases)

  $$\sum_{3r_0+2r_1+r_2=d_1+1} \binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1}\binom{d_1+1-r_0-r_1}{r_2}\frac{(d_2+3)!}{2^{r_2}6^{r_3}}\binom{n_1}{d_1+1}\binom{n_2}{d_2+3},$$

  where $r_3 = d_1+1-r_0-r_1-r_2$.

– Consider the next $\mathcal{R}(\mathcal{S})$ with size $(d_1+2)(d_2+2)$ as given in the table in Figure 4.14. We have $\beta = d_1+d_2+3$. No stopping sets are found (by contradiction on β) except for $d_2 = d_1+1$. In this case, we get $\beta = 2(d_1+2)$. The rectangle has two zeros in each row. This problem is solved in a similar method as in the proofs of Lemma 4.3 and Lemma 4.4. Indeed, we have to enumerate bipartite graphs with d_1+2 left vertices all of degree 2. These graphs have d_1+3 right vertices. Two cases should be distinguished: a- The extra vertex on the right has no edges, b- The extra vertex at the right has one edge. The number of these stopping sets is

$$\left((d_2+2)x_{d_1+2} + \frac{(d_2+2)y_{d_1+2}}{2}\right)\binom{n_1}{d_1+2}\binom{n_2}{d_2+2},$$

where x_{d_1+2} and y_{d_1+2} are determined from Lemma 4.3 and Lemma 4.4.

– The largest rectangular support for $w = (d_1+1)(d_2+1)$ is $(d_1+2)(d_2+3)$. The number of zeros is $\beta = d_2+2d_1+5$. From Lemma 4.2 we get that β must be less than or equal to both $2(d_2+3)$ and $3(d_1+2)$. The first condition is satisfied if $d_2 = d_1+1$ and $d_1 = 2$, also the second condition is satisfied if $d_2 = 2d_1-1$ and $d_1 = 2$. Consequently, for this w and this size of $\mathcal{R}(\mathcal{S})$, non-obvious stopping sets exist only for $d_1 = 2$, $d_2 = 3$, and $\beta = 12$ in a rectangle of size 4×6. Their number is

$$1860\binom{n_1}{d_1+2}\binom{n_2}{d_2+3}.$$

4.A.2 Minimum distances satisfying $d_2 = 2d_1$

- The case $d_1(d_2+1) < w < (d_1+1)d_2 = d_1(d_2+2)$.
  Write $w = d_1(d_2+1)+\lambda$, where λ is in the range $[1, d_1-1]$. For all sizes of $\mathcal{R}(\mathcal{S})$ in the table in Figure 4.14, we find $\beta = \ell_1\ell_2 - w$ and we notice that it contradicts Lemma 4.2. Thus, there are no stopping sets for w in the range $]d_1(d_2+1), (d_1+1)d_2[$.

- The case $w = (d_1+1)d_2 = d_1(d_2+2)$.

  – Obvious stopping sets do exist and their number is

  $$\binom{n_1}{d_1+1}\binom{n_2}{d_2} + \binom{n_1}{d_1}\binom{n_2}{d_2+2}.$$

  – For rectangles larger than $(d_1+1)\times d_2$ and $d_1\times(d_2+2)$, all sizes yield no stopping sets (by contradiction on β) except for $(d_1+1)\times(d_2+1)$ and $(d_1+1)\times(d_2+2)$ where the number of non-obvious stopping sets is respectively

  $$(d_1+1)!\binom{d_2+1}{d_1+1}\binom{n_1}{d_1+1}\binom{n_2}{d_2+1},$$

and

$$\frac{(d_2+2)!}{2^{d_1+1}}\binom{n_1}{d_1+1}\binom{n_2}{d_2+2}.$$

- The case $(d_1+1)d_2 = d_1(d_2+2) < w < d_1(d_2+3)$.
  Write $w = (d_1+1)d_2 + \lambda$, where λ is in the range $[1, d_1-1]$.
  - The smallest rectangular support with a non-zero number of stopping sets is $(d_1+1)\times(d_2+1)$. We have $\beta = d_1+1-\lambda$ belonging to the range $[2, d_1]$. The number of corresponding non-obvious stopping sets is
  $$(d_1+1-\lambda)!\binom{d_1+1}{\lambda}\binom{d_2+1}{d_1+\lambda}\binom{n_1}{d_1+1}\binom{n_2}{d_2+1}.$$
  - For $\mathcal{R}(\mathcal{S})$ with size $(d_1+1)\times(d_2+2)$, we have $\beta = d_2+2-\lambda$ varying in the range $[d_1+3, d_2+1]$. The rectangle have λ columns without zeros. Given $r_2 = d_1+1-r_0-r_1 = d_1+1+r_0-\lambda$, the number of non-obvious stopping sets is
  $$\sum_{2r_0+r_1=\lambda}\binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1}\frac{(d_2+2)!}{2^{r_2}\lambda!}\binom{n_1}{d_1+1}\binom{n_2}{d_2+2}.$$
  Larger rectangles $\mathcal{R}(\mathcal{S})$ lead to a contradiction on β, so they do not create stopping sets for this given weight w.

- The case $w = d_1(d_2+3)$.
  Obvious stopping sets are given by
  $$\binom{n_1}{d_1}\binom{n_2}{d_2+3}.$$
  - Take $\mathcal{R}(\mathcal{S})$ with size $(d_1+1)(d_2+1)$. Then $\beta = 1$ (recall that $d_2 = 2d_1$ in this sub-section). The number of non-obvious stopping sets with a unique zero in their rectangular support is
  $$(d_1+1)(d_2+1)\binom{n_1}{d_1+1}\binom{n_2}{d_2+1}.$$
  - Take $\mathcal{R}(\mathcal{S})$ with size $(d_1+1)(d_2+2)$. Then $\beta = d_1+2$. The number of stopping sets is given by (4.A.2) after setting $\lambda = d_1$.
  - Take $\mathcal{R}(\mathcal{S})$ with size $(d_1+1)(d_2+3)$. Then $\beta = d_2+3$. In $\mathcal{R}(\mathcal{S})$, all columns have a unique zero. Define $r_3 = d_1+1-r_0-r_1-r_2 = 2r_0+r_1+1$, then the number of non-obvious stopping sets in this sub-case becomes
  $$\sum_{3r_0+2r_1+r_2=d_1}\binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1}\binom{d_1+1-r_0-r_1}{r_2}$$
  $$\frac{(d_2+3)!}{2^{r_2}6^{r_3}}\binom{n_1}{d_1+1}\binom{n_2}{d_2+3}$$

  All remaining rectangle sizes (smaller or larger) have no stopping sets.

- For $d_2 = 2d_1$ the range $]d_1(d_2+3), (d_1+1)(d_2+1)[$ is empty. We complete this sub-section with the last case $w = (d_1+1)(d_2+1)$.
  - The number of obvious stopping sets for the smallest $\mathcal{R}(\mathcal{S})$ is
    $$\binom{n_1}{d_1+1}\binom{n_2}{d_2+1}.$$
    The size $(d_1+2)\times d_2$ rectangle has no stopping sets.
  - The next $\mathcal{R}(\mathcal{S})$ is $(d_1+1)(d_2+2)$. The corresponding number of zeros is $\beta = d_1+1$. The number of non-obvious stopping sets is (expression identical to the case $d_2 \leq 2d_1 - 1$):
    $$\sum_{2r_0+r_1=d_1+1} \binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1}\frac{(d_2+2)!}{2^{r_0}(d_1+1)!}\binom{n_1}{d_1+1}\binom{n_2}{d_2+2}.$$
  - Consider $\mathcal{R}(\mathcal{S})$ with size $(d_1+2)(d_2+1)$. We have $\beta = d_2+1$. For $d_2 = 2d_1$ this β contradicts the upper bound in Lemma 4.2. Then $\tau_w = 0$.
  - Consider $\mathcal{R}(\mathcal{S})$ with size $(d_1+1)(d_2+3)$. We have $\beta = 2(d_1+1) = d_2+2$. In $\mathcal{R}(\mathcal{S})$, all columns must have at most one zero but rows can afford up to three zeros. The number of non-obvious stopping sets is found to be (method as in previous cases)
    $$\sum_{3r_0+2r_1+r_2=d_1+1} \binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1}\binom{d_1+1-r_0-r_1}{r_2} \frac{(d_2+3)!}{2^{r_2}6^{2r_0+r_1}}\binom{n_1}{d_1+1}\binom{n_2}{d_2+3}.$$
  - Consider the next supports $\mathcal{R}(\mathcal{S})$ with size $(d_1+2)(d_2+2)$ and $(d_1+2)(d_2+3)$ as given in the table in Figure 4.14. The β for both sizes contradicts the upper bound in Lemma 4.2. We deduce that $\tau_w = 0$ in these cases.

4.A.3 Minimum distances satisfying $d_2 > 2d_1$

For $2 < d_1 < d_2 < 3d_1 - 1$, the width of $\mathcal{R}(\mathcal{S})$ cannot exceed d_2+3. In the special case $d_1 = 2$, as stated earlier, a width up to d_2+4 should be considered. Then, for $d_1 = 2$, the rectangular supports are ordered in increasing size according to Table 4.4. The first and second rows list the stopping set weight w in increasing order.

- The case $d_1(d_2+1) < w < d_1(d_2+2) < (d_1+1)d_2$.
  Write $w = d_1(d_2+1)+\lambda$, where λ is in the range $[1, d_1-1]$. For all sizes of $\mathcal{R}(\mathcal{S})$ in the table in Figure 4.14, we find $\beta = \ell_1\ell_2 - w$ and we notice that it contradicts Lemma 4.2. There are no stopping sets for w in the range $]d_1(d_2+1), d_1(d_2+2)[$.

$d_1d_2 < d_1(d_2+1) < d_1(d_2+2) < (d_1+1)d_2$ $< d_1(d_2+3) < \mathbf{d_1(d_2+4)} \leq (d_1+1)(d_2+1)$ $< (d_1+2)d_2 \leq (d_1+1)(d_2+2) < (d_1+1)(d_2+3)$ $\leq (d_1+2)(d_2+1) < (\mathbf{d_1+1})(\mathbf{d_2+4}) < (d_1+2)(d_2+2)$ $< (d_1+2)(d_2+3) < (\mathbf{d_1+2})(\mathbf{d_2+4})$

Table 4.4: Table of rectangular sizes for the special case where the first component code has $d_1 = 2$.

- The case $w = d_1(d_2+2)$.
  - Obvious stopping sets do exist and their number is
$$\binom{n_1}{d_1}\binom{n_2}{d_2+2}.$$
  - For rectangles larger than $d_1(d_2+2)$ we found no other stopping sets, by contradiction on β.
- The case $d_1(d_2+2) < w < (d_1+1)d_2$.
  Write $w = d_1(d_2+2)+\lambda$, where λ is in the range $[1, d_2-2d_1-1]$. For all rectangular supports, from β we deduce that $\tau_w = 0$.
- The case $w = (d_1+1)d_2$.
  Obvious stopping sets are given by
$$\binom{n_1}{d_1+1}\binom{n_2}{d_2}.$$
  - Take $\mathcal{R}(\mathcal{S})$ with size $(d_1+1)(d_2+1)$. Then $\beta = d_1+1$. The number of non-obvious stopping sets for this sub-case is
$$(d_1+1)!\binom{d_2+1}{d_1+1}\binom{n_1}{d_1+1}\binom{n_2}{d_2+1}.$$
  - Take $\mathcal{R}(\mathcal{S})$ with size $(d_1+1)(d_2+2)$. Then $\beta = 2d_1+2$. All rows have two zeros. Also, d_2-2d_1 columns have no zeros, while the remaining columns include a unique zero. The number of non-obvious stopping sets in this sub-case is
$$\frac{(d_2+2)!}{2^{d_1+1}(d_2-2d_1)!}\binom{n_1}{d_1+1}\binom{n_2}{d_2+2}.$$
  - Take $\mathcal{R}(\mathcal{S})$ with size $(d_1+1)(d_2+3)$. From Lemma 4.2 we find that no stopping sets exist, except for $d_1 = 2$ and $d_2 = 6$. In this case, $\beta = 3d_1+3 = 9$. Each row in the rectangle have three zeros. The number of stopping sets is $\tau_w = \frac{9!}{6^3} = 1680$ for $d_1 = 2$ and $d_2 = 6$.

  - Take $\mathcal{R}(\mathcal{S})$ with size $(d_1+2)(d_2+1)$. Then $\beta = d_2+d_1+2$. The reader can easily check that the bound in Lemma 4.2 is not satisfied. We deduce that $\tau_w = 0$ for this rectangle size and this weight w. All remaining rectangle sizes have no stopping sets.

- The case $(d_1+1)d_2 < w < d_1(d_2+3)$.
  $\tau_w = 0$ for $d_1 = 2$. We pursue this case for $d_1 > 2$.
  Write $w = (d_1+1)d_2+\lambda$ where λ belongs to the non-empty interval $[1, 3d_1-d_2-1]$.

  - The next rectangular support with a non-zero number of stopping sets is $(d_1+1)(d_2+1)$. The number of zeros is $\beta = d_1+1-\lambda$ varying in the range $[d_2-2d_1+2, d_1]$. Non-obvious stopping sets are enumerated by selecting the location and permuting the β zeros inside $\mathcal{R}(\mathcal{S})$. Their number is

$$(d_1+1-\lambda)!\binom{d_2+1}{d_1+1-\lambda}\binom{d_1+1}{\lambda}\binom{n_1}{d_1+1}\binom{n_2}{d_2+1}.$$

  - Take $\mathcal{R}(\mathcal{S})$ with size $(d_1+1)(d_2+2)$. We have $\beta = 2d_1+2-\lambda$ inside the interval $[d_2-d_1+3, 2d_1+1]$. As made before, we find $2r_0+r_1 = \lambda$ and $r_2 = d_1+1+r_0-\lambda$. The columns in $\mathcal{R}(\mathcal{S})$ have at most one zero and rows have at most two zeros. The number of these non-obvious stopping sets is

$$\sum_{2r_0+r_1=\lambda}\binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1}\frac{(d_2+2)!}{2^{r_2}(d_2-2d_1+\lambda)!}\binom{n_1}{d_1+1}\binom{n_2}{d_2+2}.$$

  - Consider $\mathcal{R}(\mathcal{S})$ with size $(d_1+2)(d_2+1)$. Here $\beta = d_2+d_1+2-\lambda$ contradicts Lemma 4.2 because $d_2 > 2d_1$. We have $\tau_w = 0$. All remaining rectangles (smaller or larger) have no stopping sets.

- The case $w = d_1(d_2+3)$.

  - The number of obvious stopping sets in a $d_1 \times (d_2+3)$ rectangular support is

$$\binom{n_1}{d_1}\binom{n_2}{d_2+3}.$$

  - Consider $\mathcal{R}(\mathcal{S})$ with size $(d_1+1)(d_2+1)$. Here $\beta = d_2-2d_1+1$ is restricted to the interval $]1, d_1[$ given the constraints $2d_1 < d_2 < 3d_1-1$ for $d_1 > 2$. For $d_1 = 2$, $d_2 = 6$ is the only valid value, with $\beta = 3$. The number of non-obvious stopping sets is (for $d_1 \geq 2$)

$$\beta!\binom{d_1+1}{3d_1-d_2}\binom{d_2+1}{2d_1}\binom{n_1}{d_1+1}\binom{n_2}{d_2+1}.$$

  - Consider $\mathcal{R}(\mathcal{S})$ with size $(d_1+1)(d_2+2)$. The number of zeros is $\beta = d_2-d_1+2 \in]d_1+2, 2d_1+1[$. The number of non-obvious stopping sets is given by

$$\sum_{2r_0+r_1=3d_1-d_2} \binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1}\frac{(d_2+2)!}{2^{r_2}d_1!}\binom{n_1}{d_1+1}\binom{n_2}{d_2+2},$$

where $r_2 = d_2 - 2d_1 + 1 + r_0$. For $d_2 = 2$, the above expression is valid for $d_2 = 6$ only.

- Consider $\mathcal{R}(\mathcal{S})$ with size $(d_1+1)(d_2+3)$. Here $\beta = d_2+3$. All columns in the rectangle have one zero. Rows can have up to three zeros. The number of non-obvious stopping sets is

$$\sum_{3r_0+2r_1+r_2=3d_1-d_2} \binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1}$$
$$\binom{d_1+1-r_0-r_1}{r_2}\frac{(d_2+3)!}{2^{r_2}6^{r_3}}\binom{n_1}{d_1+1}\binom{n_2}{d_2+3},$$

  where $r_3 = d_1 + 1 - r_0 - r_1 - r_2$. The above expression is also valid for $(d_1, d_2) = (2, 6)$.
- The rectangular support $\mathcal{R}(\mathcal{S})$ of size $(d_1+2)(d_2+1)$ gives no stopping sets. The remaining rectangular supports from Figure 4.14 and Table 4.4 yield no stopping sets for $w = d_1(d_2+3)$.

Recall that a maximal rectangle width of d_2+3 should be considered for $d_1 > 2$ and it goes up to d_2+4 for $d_1 = 2$ as shown in Table 4.4. New obvious stopping sets are found, they appear for $d_1 = 2$ only with a rectangular width equal to d_2+4. Their rectangular support corresponds to the sizes in boldface in Table 4.4 for $w > d_1(d_2+3)$: $\binom{n_1}{d_1}\binom{n_2}{d_2+4}$ obvious stopping sets of size $d_1 \times (d_2+4)$, $\binom{n_1}{d_1+1}\binom{n_2}{d_2+4}$ obvious stopping sets of size $(d_1+1)\times(d_2+4)$, and $\binom{n_1}{d_1+2}\binom{n_2}{d_2+4}$ obvious stopping sets of size $(d_1+2)\times(d_2+4)$. Given that this theorem enumerates stopping sets for $w \leq (d_1+1)(d_2+1)$, one should only count obvious $d_1 \times (d_2+4)$ sets.

- The case $d_1(d_2+3) < w < (d_1+1)(d_2+1)$.
  Write $w = d_1(d_2+3)+\lambda$ where $\lambda \in [1, d_2-2d_1]$. The results for the three rectangles listed below are valid for $d_1 \geq 2$.
  - Consider $\mathcal{R}(\mathcal{S})$ of size $(d_1+1)(d_2+1)$. We have $\beta = d_2 - 2d_1 + 1 - \lambda$ varying in the range $[1, d_2-2d_1]$. The number of non-obvious stopping sets is

$$(d_2-2d_1+1-\lambda)!\binom{d_1+1}{3d_1-d_2+\lambda}\binom{d_2+1}{2d_1+\lambda}\binom{n_1}{d_1+1}\binom{n_2}{d_2+1}.$$

  - Now consider $\mathcal{R}(\mathcal{S})$ of size $(d_1+1)(d_2+2)$. We have $\beta = d_2 - d_1 + 2 - \lambda \in [d_1+2, d_2-d_1+1]$. As done before, the expression of the number of non-obvious stopping sets involves r_0 and r_1 as follows.

$$\sum_{2r_0+r_1=3d_1-d_2+\lambda} \binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1}\frac{(d_2+2)!}{2^{r_2}(d_1+\lambda)!}\binom{n_1}{d_1+1}\binom{n_2}{d_2+2},$$

where $r_2 = d_1 + 1 - r_0 - r_1$.

- We consider the next $(d_1+1)(d_2+3)$ rectangular support. Now $\beta = d_2+3-\lambda \in [2d_1+3, d_2+2]$. The number of non-obvious stopping sets is

$$\sum_{3r_0+2r_1+r_2=3d_1-d_2+\lambda} \binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1} \binom{d_1+1-r_0-r_1}{r_2}\frac{(d_2+3)!}{2^{r_2}6^{r_3}\lambda!}\binom{n_1}{d_1+1}\binom{n_2}{d_2+3},$$

  where $r_3 = d_1 + 1 - r_0 - r_1 - r_2$.

- The remaining larger rectangular supports give no stopping sets, except for $(d_1+1)(d_2+4)$ for $d_1 = 2$ and $d_2 = 6$ where $\tau_w = 22050$. These 22050 rectangles of size 3×10, where $\beta = 10$ and $w = 20$, have one zero in each column but a row may have up to four zeros.

- The last case $w = (d_1+1)(d_2+1)$.

  - Obvious stopping sets are given by

$$\binom{n_1}{d_1+1}\binom{n_2}{d_2+1}.$$

  - The next $\mathcal{R}(\mathcal{S})$ from the table is $(d_1+1)\times(d_2+2)$. We have $\beta = d_1+1$. The number of stopping sets is

$$(d_1+1)!\binom{d_2+2}{d_1+1}\binom{n_1}{d_1+1}\binom{n_2}{d_2+2}.$$

  - Now consider the $(d_1+1)\times(d_2+3)$ rectangle. We have $\beta = 2d_1+2$. The number of stopping sets is

$$\sum_{3r_0+2r_1+r_2=d_1+1} \binom{d_1+1}{r_0}\binom{d_1+1-r_0}{r_1}\binom{d_1+1-r_0-r_1}{r_2} \frac{(d_2+3)!}{2^{r_2}6^{r_3}(d_2-2d_1+1)!}\binom{n_1}{d_1+1}\binom{n_2}{d_2+3},$$

  where $r_3 = d_1 + 1 - r_0 - r_1 - r_2$, the above expression being valid for $d_1 \geq 2$.

– No stopping sets are found for the remaining three rectangular supports for $d_1 > 2$. On the other hand, for $d_1 = 2$, stopping sets are found only with a rectangle $(d_1+1)(d_2+4)$. In this case, we have $\beta = 3(d_1+1)$. The number of non-obvious stopping sets is $\tau_w = 11130$ for $(d_1, d_2) = (2, 5)$ and $\tau_w = 111300$ for $(d_1, d_2) = (2, 6)$.

Q.E.D.

Bibliography

[1] N. Abramson, "Cascade decoding of cyclic product codes," *IEEE Trans. on Comm. Technology*, vol. 16, no. 3, pp. 398-402, June 1968.

[2] M. Alipour, O. Etesami, G. Maatouk, and A. Shokrollahi, "Irregular product codes," *IEEE Information Theory Workshop*, pp 197-201, Lausanne, Sept. 2012.

[3] M. Asteris and A. G. Dimakis, "Repairable fountain codes," *IEEE Proceedings of ISIT*, 2012.

[4] D. Augot, M. El-Khamy, R.J. McEliece, F. Parvaresh, M. Stepanov, and A. Vardy, "Algebraic list decoding of Reed-Solomon product codes," *Algebraic and Combinatorial Coding Workshop*, pp. 210-213, Sept. 2006.

[5] E. Arıkan, "Channel polarization: A method for constructing capacity achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. on Inf. Theory*, vol. 55, no. 7, pp. 3051-3073, July 2009.

[6] S. Benedetto and G. Montorsi, "Unveiling turbo-codes: some results on parallel concatenated coding schemes," *IEEE Trans. on Inf. Theory*, vol. 42, no. 2, pp. 409-428, March 1996.

[7] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: turbo codes," *Proc. International Conference on Communications (ICC)*, pp. 1064-1070, Geneva, 1993.

[8] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-codes," *IEEE Trans. on Communications*, vol. 44, pp. 1261-1271, Oct. 1996.

[9] E. Biglieri, *Coding for Wireless Channels*, Springer, 2005.

[10] R.E. Blahut, *Algebraic codes for data transmission*, Cambridge University Press, 2003.

[11] B. Bollobás, *Modern graph theory*, Springer, 1998.

[12] R.C. Bose and D.K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Information and Control*, vol. 3, pp. 68–79, 279-290, March 1960.

[13] J.J. Boutros, O. Pothier, and G. Zémor, "Generalized low density (Tanner) codes," *IEEE Intern. Conf. on Comm. (ICC)*, vol. 1, pp. 441-445, Vancouver, June 1999.

[14] J.J. Boutros, G. Zémor, A. Guillén i Fàbregas, and E. Biglieri, "Full-diversity product codes for block erasure and block fading channels," *Information Theory Workshop*, pp. 313-317, Porto, May 2008.

[15] J.J. Boutros, G. Zémor, A. Guillén I Fàbregas, and E. Biglieri, "Generalized low-density codes with BCH constituents for full-diversity near-outage performance," *IEEE International Symposium on Information Theory (ISIT)*, pp. 787-791, Toronto, July 2008.

[16] J.J. Boutros, "Diversity and coding gain evolution in graph codes," *Information Theory and Applications Workshop*, pp. 34-43, Feb. 2009.

[17] J.J. Boutros, A. Guillén i Fàbregas, E. Biglieri, and G. Zémor, "Low-density parity-check codes for nonergodic block-fading channels," *IEEE Trans. on Inf. Theory*, vol. 56, no. 9, pp. 4286-4300, Sept. 2010.

[18] J. Boutros, V. Dedeoglu, and M. Bloch, "The anti-diversity concept for secure communication on a two-link compound channel," *in Proc. International Zurich Seminar on Communications (IZS)*, pp. 116-119, Feb. 2014.

[19] A.R. Calderbank and N. Seshadri, "Multilevel codes for unequal error protection," *IEEE Trans. on Inf. Theory*, vol. 39, no. 4, pp. 1234-1248, July 1993.

[20] E.R. Canfield and B.D. McKay, "Asymptotic enumeration of integer matrices with constant row and column sums", Combinatorics, arXiv:math/0703600, revised June 2009.

[21] J. Chakareski, S. Han, and B. Girod, "Layered coding vs multiple descriptions for video streaming over multiple paths," *ACM international conference on Multimedia*, pp. 422-431, Berkeley, CA, 2003.

[22] J.H. Conway and R.K. Guy. *The Book of Numbers*. New York: Springer-Verlag, pp. 94-96, 1996.

[23] M. Lentmaier, A. Sridharan, K.Sh. Zigangirov, and D.J. Costello, "Terminated LDPC convolutional codes with thresholds close to capacity", *IEEE Int. Symposium on Inform. Theory*, pp. 1372-1376, Sept. 2005.

[24] C. Greenhill and B.D. McKay, "Asymptotic enumeration of sparse nonnegative integer matrices with specified row and column sums," Advances in Applied Mathematics, vol. 41, pp. 459-481, 2008, revised April 2012.

[25] A. Datta, and F. Oggier, "An overview of codes Tailor-made for networked distributed data storage," *CoRR*, 2011.

[26] V. Dedeoglu and J.J. Boutros, "Diversity-Security tradeoff for compound channels," *in IEEE International Conference on Communications (ICC)*, London, June 2015.

[27] C. 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," *IEEE Trans. on Inf. Theory*, vol. 48, no. 6, pp. 1570-1579, Jun. 2002.

[28] A. G. Dimakis, B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage system," *IEEE Transactions on Information Theory*, vol. 56, pp. 4539–4551, 2010.

[29] A.G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A survey on network codes for distributed storage," *IEEE Proceedings*, vol. 99, pp. 476-489, March 2011.

[30] Q. Du and X. Zhang, "On rate adaptation for video multicast with layered coding over multirate wireless networks," *IEEE Global Telecommunications Conference*, pp. 1-5, New Orleans, Nov. 2008.

[31] P. Elias, "Error-free coding," *IRE Trans. Inf. Theory*, vol. 4, no. 4, pp. 29-39, Sept. 1954.

[32] M. El-Khamy and R.J. McEliece, "Iterative algebraic soft-decision list decoding of Reed-Solomon codes," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 3, pp. 481-490, March 2006.

[33] K.S. Esmaili, L. Pamies-Juarez, and A. Datta, "CORE: Cross-object redundancy for efficient data repair in storage systems," *IEEE Big Data*, pp. 246-254, Oct. 2013.

[34] A.J. Felström and K.S. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix," *IEEE Trans. on Inf. Theory*, vol. 45, no. 6, pp. 2181-2191, Sept. 1999.

[35] D.F. Freeman and A.M. Michelson, "A two-dimensional product code with robust soft-decision decoding," *IEEE Trans. Comm.*, vol. 44, no. 10, pp. 1222-1226, Oct. 1996.

[36] R.G. Gallager, *Low-density parity-check codes*, Ph.D. thesis, Massachussets Institute of Technology Press, 1963.

[37] M. Ghanbari, "Two-layer coding for video signals for VBR networks," *IEEE Journal in Selected Areas in Communications*, vol. 7, no. 5, pp. 771-781, June 1989.

[38] P. Gopalan, A.R. Klivans, and D. Zuckerman, "List-decoding Reed–Muller codes over small fields," *40th Annu. ACM Symp. Theory of Computing (STOC)*, pp. 265–274, New York, 2008.

[39] P. Gopalan, V. Guruswami, and P. Raghavendra, "List decoding tensor products and interleaved codes," *SIAM J. Comput.*, vol. 40, no. 5, pp. 1432-1462, Oct. 2011.

[40] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, "On the locality of codeword symbols," *IEEE Trans. on Inf. Theory*, vol. 58, no. 11, pp. 6925-6934, Nov. 2012.

[41] A. Guillén i Fàbregas, "Coding in the block-erasure channel," *IEEE Trans. on Inf. Theory*, vol. 52, no. 11, pp. 5116-5121, Nov. 2006.

[42] A. Guillén i Fàbregas and G. Caire, "Coded modulation in the block-fading channel: Coding theorems and code construction," *IEEE Trans. on Inf. Theory*, vol. 52, no. 1, pp. 91-114, Jan. 2006.

[43] V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon and algebraic-geometry codes," *IEEE Trans. on Inf. Theory*, vol. 45, no. 6, pp. 1757–1767, June 1999.

[44] T. Helleseth, T. Klove, and O. Ytrehus, "Generalized Hamming weights of linear codes," *IEEE Trans. on Inf. Theory*, vol. 38, no. 3, pp. 1133-1140, May 1992.

[45] W. Henkel, K. Hassan, N. von Deetzen, S. Sandberg, L. Sassatelli, and D. Declercq, "UEP concepts in modulation and coding," *Hindawi Publishing, Advances in Multimedia*, article ID 416797, 2010.

[46] A. Hocquenghem, "Codes correcteurs d'erreurs," *Chiffres (Paris)*, vol. 2, pp. 147–156, in French, Sept. 1959.

[47] P. Huang, E. Yaakobi, H. Uchikawa, and P.H. Siegel, "Cyclic linear binary locally repairable codes," *IEEE Information Theory Workshop (ITW)*, pp. 1-5, April 2015.

[48] J. Jiang, K.R. Narayanan, "Iterative soft decoding of Reed-Solomon codes," *IEEE Communications Letters*, vol. 8, no. 4, pp. 244-246, April 2004.

[49] J. Katz and L. Trevisan, "On the efficiency of local decoding procedures for error-correcting codes," *32nd ACM Symposium on Theory of Computing (STOC)*, pp. 80-86, New York, 2000.

[50] S.B. Korada, E. Sasoglu, and R. Urbanke, "Polar codes: Characterization of exponent, bounds, and constructions," *IEEE Trans. on Inf. Theory*, vol. 56, no. 12, pp. 6253-6264, Dec. 2010.

[51] R. Knopp and P.A. Humblet, "On coding for block fading channels," *IEEE Trans. on Inf. Theory*, vol. 46, no. 1, pp. 189-205, Jan. 2000.

[52] F.R. Kschischang, Product Codes, J.G. Proakis (ed), *Wiley encyclopedia of telecommunications*, pp. 2007-2012, vol. 4, Hoboken, NJ, 2003.

[53] F.R. Kschischang, B.J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. on Inf. Theory*, vol. 47, pp. 498-519, Feb. 2001.

[54] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi and S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, "Oceanstore: An architecture for global-scale persistent storage," *9th Int. Conf. on Architectural Support Programm*, pp. 190-201, Cambridge, Massachusetts, 2000.

[55] S. Kudekar, T. Richardson, and R.L. Urbanke, "Threshold saturation via spatial coupling: why convolutional LDPC ensembles perform so well over the BEC," *IEEE Trans. on Inf. Theory*, vol. 57, no. 2, pp. 803-834, Feb. 2011.

[56] S. Kudekar, T. Richardson, and R.L. Urbanke, "Spatially coupled ensembles universally achieve capacity under belief propagation," *IEEE Trans. on Inf. Theory*, vol. 59, no. 12, pp. 7761-7813, Dec. 2013.

[57] S. Kudekar, M. Mondelli, E. Sasoglu, and R. Urbanke, "Reed-Muller codes achieve capacity on the binary erasure channel under MAP decoding," ArXiv 1505.05831, 2015.

[58] S. Kumar and H.D. Pfister, "Reed-Muller codes achieve capacity on erasure channels," ArXiv 1505.05123, 2015.

[59] A. Lapidoth, "Convolutional codes and finite interleavers for the block erasure channel," *Mobile Communications Advanced Systems and Components*, Lecture Notes in Computer Science, Springer Berlin Heidelberg, vol. 783, pp. 113-120, May 2005.

[60] M. Lentmaier and G. P. Fettweis, "On the thresholds of generalized LDPC convolutional codes based on protographs," *IEEE Int. Symposium on Inf. Theory*, Austin, USA, pp. 709-713, Austin, June 2010.

[61] S. Lin and D.J. Costello, *Error control coding*, Prentice Hall, 2nd edition, 2004.

[62] M. Luby, M. Mitzenmacher, M.A. Shokrollahi, D.A. Spielman, and V. Stemann, "Practical loss-resilient codes," *29th ACM Annual Symp. Theory of Computing (STOC'97)*, pp. 150–159, 1997.

[63] M. Luby, "LT codes," *IEEE 43rd Annual Symposium on Foundations of Computer Science*, pp. 271-282, Nov. 2002.

[64] David J.C. MacKay, "Information theory, inference, and learning algorithms," Cambridge University Press, 2003.

[65] F.J. MacWilliams and N.J.A. Sloane, *The theory of error-correcting codes*, North-Holland, 1977.

[66] E. Malkamaki and H. Leib, "Evaluating the performance of convolutional codes over block fading channels," *IEEE Trans. on Inf. Theory*, vol. 45, no. 5, pp. 1643-1646, July 1999.

[67] B. Masnick and J.K. Wolf, "On linear unequal error protection codes," *IEEE Trans. on Inf. Theory*, vol. 13, no. 4, pp. 600-607, Oct. 1967.

[68] D.E. Muller, "Application of Boolean algebra to switching circuit design and to error detection," *Trans. on Electronic Computers of the I.R.E. Professional Group*, vol. EC-3, no. 3, pp. 6-12, Sept. 1954.

[69] F. Oggier and A. Datta, "Self-repairing homomorphic codes for distributed storage systems," *The 30th IEEE International Conference on Computer Communications*, Infocom 2011.

[70] F. Oggier and A. Datta, "Coding techniques for repairability in networked distributed storage systems," *Foundations and Trends in Communications and Information Theory*, vol. 9, pp. 383-466, 2013.

[71] G.C. Onwubolu and D. Davendra, *Differential evolution: a handbook for global permutation-based combinatorial optimization*, Springer, 2009.

[72] R. Pellikaan and Xin-Wen Wu, "List decoding of q-ary Reed-Muller codes," *IEEE Trans. on Inf. Theory*, vol. 50, no. 4, pp. 679-682, April 2004.

[73] H. Pishro-Nik, N. Rahnavard, and F. Fekri, "Nonuniform error correction using low-density parity-check codes," *IEEE Trans. on Inf. Theory*, vol. 51, no. 7, pp. 2702-2714, July 2005.

[74] Y. Polyanskiy, H.V. Poor, and S. Verdú, "Channel coding rate in the finite block-length regime," *IEEE Trans. on Inf. Theory*, vol. 56, no. 5, pp. 2307-2359, May 2010.

[75] J.G. Proakis and M. Salehi. *Digital Communications*, McGraw-Hill, 5th edition, 2008.

[76] R. Pyndiah, "Near-optimum decoding of product codes: Block turbo codes," *IEEE Trans. Comm.*, vol. 46, no. 8, pp. 1003-1010, Aug. 1998.

[77] D. Rankin and T.A. Gulliver, "Asymptotic performance of product codes," *IEEE International Conference on Communications*, pp. 431-435, Vancouver, June 1999.

[78] K.V. Rashmi, Nihar B. Shah, P. Vijay Kumar, and K. Ramchandran, "Explicit construction of optimal exact regenerating codes for distributed storage," *Proceedings of Allerton Conference on Communication, Control and Computing*, Sept. 2009.

[79] K.V. Rashmi, Nihar B. Shah, and P. Vijay Kumar, "Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction," *IEEE Trans. on Inf. Theory*, vol. 57, no. 8, pp. 5227- 5239, Aug. 2011.

[80] K.V. Rashmi, N.B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A solution to the network challenges of data recovery in erasure-coded distributed storage systems: A study on the Facebook warehouse cluster," *Proc. USENIX HotStorage*, June 2013.

[81] V. Rathi, R. Urbanke, M. Andersson, and M. Skoglund, "Rate-equivocation optimal spatially coupled LDPC codes for the BEC wiretap channel," *IEEE Int. Symposium on Inf. Theory*, pp. 2393-2397, July 2011.

[82] S.R. Reddy and J.P. Robinson, "Random Error and Burst Correction by Iterated Codes," *IEEE Trans. on Inf. Theory*, vol. 18, no. 1, pp. 182-185, Jan. 1972.

[83] I.S. Reed, "A class of multiple-error-correcting codes and the decoding scheme," *Trans. on Inf. Theory of the IRE Professional Group* , vol. 4, no. 4, pp. 38-49, Sept. 1954.

[84] I.S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial an Applied Mathematics*, no. 8, pp. 300-304, 1960.

[85] T. Richardson, "Multi-Edge Type LDPC Codes," *presented at the Workshop honoring Prof. Bob McEliece on his 60th birthday (but not included in the proceedings)*, California Institute of Technology, Pasadena, California, May 2002.

[86] T.J. Richardson and R.L. Urbanke, *Modern coding theory*, Cambridge University Press, 2008.

[87] E. Rosnes and O. Ytrehus, "Turbo decoding on the binary erasure channel: Finite-length analysis and turbo stopping sets," *IEEE Trans. on Inf. Theory*, vol. 53, no. 11, pp. 4059-4075, Nov. 2007.

[88] E. Rosnes, "Stopping set analysis of iterative row-column decoding of product codes," *IEEE Trans. on Inf. Theory*, vol. 54, no. 4, pp. 1551-1560, April 2008.

[89] W.E. Ryan and S. Lin, *Channel codes: Classical and modern*, Cambridge University Press, 2009.

[90] A. Sarwate, "Soft decision decoding of Reed-Solomon product codes," *EECS 229B Final Project Report*, May 2005.

[91] M. Sathiamoorthy, M. Asteris, D.S. Papailiopoulos, A.G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, "XORing Elephants: Novel Erasure Codes for Big Data," *Proc. VLDB Endow*, March 2013.

[92] M. Schwartz, P.H. Siegel, and A. Vardy, "On the asymptotic performance of iterative decoders for product codes," *IEEE International Symposium on Information Theory*, pp. 1758-1762, Sept. 2005.

[93] M. Schwartz and A. Vardy, "On the stopping distance and the stopping redundancy of codes," *IEEE Trans. on Inf. Theory*, vol. 52, no. 3, pp. 922-932, March 2006.

[94] A. Sella and Y. Be'ery, "Convergence analysis of turbo decoding of product codes," *IEEE Trans. on Inf. Theory*, vol. 47, no. 2, pp. 723-735, Feb. 2001.

[95] N. Sendrier, "Codes correcteurs d'erreurs à haut pouvoir de correction," Thèse de Doctorat de l'Université Paris 6, in French, Dec. 1991.

[96] A. Shokrollahi, "Raptor codes," *IEEE Trans. on Communications*, vol. 52, no. 6, pp. 2551-2567, June 2006.

[97] N.J.A Sloane, The On-Line Encyclopedia of Integer Sequences. See sequence A000041 at oeis.org/A000041.

[98] R.P. Stanley, *Enumerative combinatorics*, Cambridge Univ. Press, vol. 1, 2nd edition, 2012.

[99] R. Storn and K. Price, "Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341-359, 1997.

[100] I. Tal and A. Vardy, "List Decoding of Polar Codes," *IEEE Trans. on Inf. Theory*, vol. 61, no. 5, pp. 2213-2226, May 2015.

[101] R.M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. on Inf. Theory*, vol. 27, no. 5, pp. 533-547, Sept. 1981.

[102] R.M. Tanner, D. Sridhara, A. Sridharan, T.E. Fuja, and D.J. Costello, "LDPC block and convolutional codes based on circulant matrices", *IEEE Trans. on Inf. Theory*, vol. 50, no. 12, pp. 2966-2984, Dec. 2004.

[103] P.S. Chandrashekhar Thejaswi, A. Bennatan, J. Zhang, A.R. Calderbank, and D. Cochran, "Layered coding for interference channels with partial transmitter side information," *IEEE Trans. on Inf. Theory*, vol. 57, no. 5, pp. 2765-2780, May 2011.

[104] J.-P. Tillich and G. Zémor, "Optimal cycle codes constructed from Ramanujan graphs," *SIAM J. Discrete Mathematics*, vol. 10, no. 3, pp. 447-459, 1997.

[105] L.M.G.M. Tolhuizen, "More results on the weight enumerator of product codes," *IEEE Trans. on Inf. Theory*, vol. 48, no. 9, pp. 2537-2577, Sept. 2002.

[106] D. Truhachev, D.G.M Mitchell, M. Lentmaier, and D.J. Costello, "New codes on graphs constructed by connecting spatially coupled chains," *Information Theory and Applications (ITA'2012)*, San Diego, CA, pp. 392-397, Feb. 2012.

[107] D. Truhachev, D.G.M Mitchell, M. Lentmaier, and D.J. Costello, "New codes on graphs constructed by connecting spatially coupled chains," arXiv:1312.3368, Dec. 2013.

[108] D.N.C. Tse and P. Viswanath, *Fundamentals of Wireless Communication*, Cambridge University Press, 2005.

[109] W.M.C.J. Van Overveld, "Multiple-burst error-correcting cyclic product codes (Corresp.)," *IEEE Trans. on Inf. Theory*, vol. 33, no. 6, pp. 919-923, Nov. 1987.

[110] D.P. Varodayan, "Investigation of the Elias product code construction for the binary erasure channel", B.A.S. Thesis, University of Toronto, Dec. 2002.

[111] S. Vialle and J.J. Boutros, "A Gallager-Tanner construction based on convolutional codes," *Workshop on Coding and Cryptography (WCC'99)*, pp. 393-404, Paris, France, Jan. 1999.

[112] A.J. Viterbi and J.K. Omura, *Principles of Digital Communication and Coding*, McGraw-Hill, 1979.

[113] S. Wainberg, "Error-erasure decoding of product codes (Corresp.)," *IEEE Trans. on Inf. Theory*, vol. 18, no. 6, pp. 821-823, Nov. 1972.

[114] V.K. Wei, "Generalized Hamming weights for linear codes," *IEEE Trans. on Inf. Theory*, vol. 37, no. 5, pp. 1412-1418, Sept. 1991.

[115] L.-J. Weng and G. Sollman, "Variable redundancy product codes," *IEEE Trans. on Comm. Technology*, vol. 15, no. 6, pp. 835-838, Dec. 1967.

[116] S.B. Wicker and V.K. Bhargava, eds., *Reed-Solomon Codes and their Applications.* New York: IEEE Press, 1994.

[117] J.K. Wolf, "On codes derivable from the tensor product of check matrices," *IEEE Trans. on Inf. Theory*, vol. 11, no. 2, pp. 281-284, April 1965.

[118] A. Yedla, P.S. Nguyen, H.D. Pfister, and K.R. Narayanan, "Universal codes for the gaussian MAC via spatial coupling," *49th Annual Allerton Conf. on Commun., Control, and Computing*, pp. 1801-1808, Sept. 2011.

[119] S. Yekhanin, "Locally decodable codes.," *Computer Science – Theory and Applications*, vol. 6651 of the series Lecture Notes in Computer Science, pp. 289-290, 2012.

Publications

Journal Papers

1. **Fanny Jardel** and Joseph J. Boutros, "Edge Coloring in Product Code Graphs for Distributed Storage," revised to the IEEE Transactions on Information Theory, Dec. 2015.

Conferences

1. **Fanny Jardel**, Joseph J. Boutros, and Mireille Sarkiss, "Stopping sets for MDS-based product codes," *IEEE International Symposium on Information Theory (ISIT)*, Barcelona, July 2016.

2. **Fanny Jardel**, J.J. Boutros, V. Dedeoglu, M. Sarkiss, and G. Rekaya-Ben Othman, "Spatial coupling for distributed storage and diversity applications," *IEEE International Conference in Communications and Networking (ComNet)*, Hammamet, Nov. 2015.

3. V. Dedeoglu, **Fanny Jardel** and J. Boutros, "Spatial coupling of root-LDPC: Parity bits doping," *IEEE International Conference on Telecommunications (ICT)*, pp. 272-276, Sydney, April 2015.

4. **Fanny Jardel** and J.J. Boutros, "Non-uniform spatial coupling," *IEEE Information Theory Workshop (ITW)*, pp. 446-450, Hobart, Nov. 2014.

Calcul et Stockage Distribués pour les Réseaux de Communication

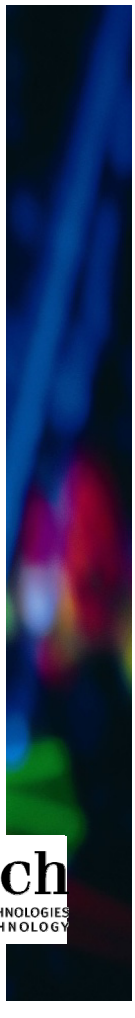
RÉSUMÉ : Ce travail est dédié à la conception, l'analyse et l'évaluation des performances de nouveaux schémas de codage appropriés aux systèmes de stockage distribué. La première partie de ce travail est consacrée à l'étude des performances des codes spatialement couplés pour les canaux à effacements. Une nouvelle méthode de couplage spatial des ensembles classiques de contrôle de parité à faible densité (LDPC) est proposée. La méthode est inspirée du codage en couches. Les arêtes des ensembles locaux et celles définissant le couplage spatial sont construites séparément. Nous proposons également de saturer le seuil d'un ensemble Root-LDPC par couplage spatial de ses bits de parité dans le but de faire face aux évanouissements quasi-statiques. Le couplage spatial est dans un deuxième temps appliqué à un ensemble Root-LDPC, ayant une double diversité, conçu pour un canal à effacements par blocs à 4 états. Dans la deuxième partie de ce travail, nous considérons les codes produits non-binaires avec des composantes MDS et leur décodage algébrique itératif ligne-colonne sur un canal à effacements. Les effacements indépendants et par blocs sont considérés. Une représentation graphique compacte du code est introduite avec laquelle nous définissions la notion de coloriage à double diversité. Les ensembles d'arrêt sont définis et une caractérisation complète est donnée. La performance des codes produits à composantes MDS, avec et sans coloration, à double diversité est analysée en présence d'effacements indépendants et par blocs. Les résultats numériques montrent aussi une excellente performance en présence d'effacements à probabilité inégale due au coloriage ayant une double diversité.

Mots-Clés : Stockage distribué, codes LDPC, codes Root-LDPC, codes produits, couplage spatial, coloration d'arêtes, ensemble d'arrêt, diversité, décodage itératif, canal à effacements.

Distributed Coding and Computing for Networks

ABSTRACT : This work is dedicated to the design, analysis, and the performance evaluation of new coding schemes suitable for distributed storage systems. The first part is devoted to spatially coupled codes for erasure channels. A new method of spatial coupling for low-density parity-check ensembles is proposed. The method is inspired from overlapped layered coding. Edges of local ensembles and those defining the spatial coupling are separately built. We also propose to saturate the whole Root-LDPC boundary via spatial coupling of its parity bits to cope with quasi-static fading. Then, spatial coupling is applied on a Root-LDPC ensemble with double diversity designed for a channel with 4 block-erasure states. In the second part of this work, we consider non-binary product codes with MDS components and their iterative row-column algebraic decoding on the erasure channel. Both independent and block erasures are considered. A compact graph representation is introduced on which we define double-diversity edge colorings via the rootcheck concept. Stopping sets are defined and a full characterization is given in the context of MDS components. A differential evolution edge coloring algorithm that produces colorings with a large population of minimal rootcheck order symbols is presented. The performance of MDS-based product codes with and without double-diversity coloring is analyzed in presence of both block and independent erasures. Furthermore, numerical results show excellent performance in presence of unequal erasure probability due to double-diversity colorings.

Keywords : Distributive storage, LDPC codes, Root-LDPC codes, product codes, spatial coupling, edge coloring, diversity, stopping set, iterative decoding, erasure channel.



TELECOM ParisTech

EDITE

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY