Distance properties of polar codes : theory and applications

Malek Ellouze

université de BORDEAUX

THÈSE PRÉSENTÉE
POUR OBTENIR LE GRADE DE

DOCTEUR DE L'UNIVERSITÉ DE BORDEAUX

École Doctorale n° 209 : Sciences physiques et de l'Ingénieur
Spécialité : Électronique

par **Malek Ellouze**

Propriétés de distance des codes polaires : théorie et applications

Co-directeurs de thèse : Christophe Jégo
Charly Poulliat

préparée au Laboratoire de l'Intégration du Matériau au Système (IMS)

soutenue le 5 juillet 2024

Membres du jury :

Charbel Abdel Nour	- Professeur	- IMT Atlantique	*Rapporteur*
Philippe Mary	- Professeur des universités	- INSA Rennes	*Rapporteur*
Iryna Andriyanova	- Professeur des universités	- Université Cergy Paris	*Présidente*
Stephan ten Brink	- Full professor	- Université de Stuttgart	*Examinateur*
Christophe Jégo	- Professeur des universités	- Bordeaux INP	*Co-directeur*
Charly Poulliat	- Professeur des universités	- Toulouse INP	*Co-directeur*

Membres invités :

Romain Tajan	- Maître de conférences	- Bordeaux INP	*Co-encadrant*
Camille Leroux	- Maître de conférences, HDR	- Bordeaux INP	*Co-encadrant*

Thèse réalisée au

Laboratoire de l'Intégration du Matériau au Système (IMS)
de Bordeaux, au sein de l'équipe CSN du groupe Conception.

Université de Bordeaux, Laboratoire IMS
UMR 5218 CNRS - Bordeaux INP
351 Cours de la Libération
Bâtiment A31
33405 Talence Cedex
FRANCE

Acknowledgments

Before exploring the various scientific aspects detailed in this manuscript, I wish to express my profound gratitude to the individuals whose support and guidance have been instrumental in the completion of this work. None of this would have been possible without their encouragement, insightful advice, and constant belief in me.

This work is primarily a result of the exceptional supervision I received. My deepest gratitude goes to Romain, whose dedication and countless hours of discussion were invaluable. His human qualities and belief in me, even when I struggled to believe in myself, have been a cornerstone of this journey. Camille, I am immensely grateful for the fruitful exchanges we had. Your meticulous explanations and invaluable feedback have been crucial in shaping this work. Christophe, thank you for your leadership throughout this PhD. Your rigor, unfailing support, and patience have been indispensable. Charly, your valuable advice and support have been much appreciated. All of you introduced me to the world of coding theory at this early stage of my career and fostered a deep passion for this field.

I also extend my heartfelt thanks to the members of the jury. I want to thank Prof. Iryna Andriyanova, full professor at Cergy Paris university for the honor of presiding over the jury. I am also grateful for Prof. Charbel Abdel Nour, professor at IMT atlantique and for and Prof. Philippe Mary, full professor at INSA Rennes, for their thorough review of my manuscript. I am also deeply thankful for Prof. Dr.-Ing Stephan ten Brink, full professor at the university of Stuttgart, for his examination of this work.

Furthermore, I wish to express my profound appreciation to all my colleagues from the CSN group at the IMS laboratory. The positive atmosphere within the team made working in such an environment a true pleasure. The academic discussions or simply the laughs shared have been invaluable.

During my years as a PhD student, I had the opportunity to teach for the first three

years as a part-time lecturer and in the fourth year as a Temporary Teaching and Research Assistant (ATER). I want to express my gratitude to everyone who made this experience highly enjoyable and ensured it was conducted under the best possible conditions. This teaching experience made me discover a new world, and I learned a lot from it.

Since my educational journey began long before my PhD, I would like to extend my gratitude to the teachers who crossed my path from primary school through to middle school, high school, preparatory classes, and engineering school. They instilled in me values and knowledge that have guided my present choices.

I want to express my profound appreciation to my family: my parents Hela and Hassib and my sister Farah, who have played an indispensable role in shaping the person I am today. Words cannot adequately convey how lucky and grateful I feel to have their love and unconditional support. They have been my pillars of strength during the most challenging times, always believing in me when I struggled to believe in myself. This work is not solely mine; it is also a reflection of their encouragement and guidance. I sincerely hope that they take pride in my accomplishments, as they have contributed immensely to this journey. Mom, dad, sister, this is also your achievement. I am also deeply thankful to all my family members for constantly checking on me and encouraging me in every possible ways.

In the words of William Shakespeare, "I am wealthy in my friends," and indeed, their presence has enriched this experience beyond measure. Their unwavering support over the past few years has been a treasure beyond price. While I refrain from mentioning names, as they undoubtedly know who they are, I extend my heartfelt gratitude to each of them. To my dear friends, your impact during this experience has been profound. From lifting my spirits with laughter during moments of darkness to offering company during times of need, from engaging in stimulating conversations to providing invaluable guidance, from encouraging me to reach for the stars to simply being there to share in life's joys and sorrows, your friendship has been a source of strength and joy.

Acknowledging that a PhD journey encompasses not only scientific challenges but also personal growth, I extend my heartfelt gratitude to Mrs. Meunier, my therapist. In her supportive environment, she provided me with the space and guidance needed to navigate the challenges of this journey while also facilitating my personal development.

Lastly, I wish to convey my gratitude to all those whose small gestures have positively influenced this journey. Your presence and support made a profound impact, and I am genuinely thankful for each and every one of you.

Résumé

Les codes correcteurs d'erreurs sont essentiels pour garantir des transmissions de données fiables, surtout dans des contextes où diverses interférences peuvent compromettre l'intégrité des informations. Les codes polaires sont l'une des familles de codes correcteurs d'erreurs les plus compétitives. Ils peuvent atteindre la capacité du canal de Shannon grâce à un encodage et un décodage efficaces pour de très grandes tailles de codes. Pour ces raisons, les codes polaires ont été inclus dans le standard 5G. De plus, ils sont l'objet de plusieurs recherches pour le futur standard 6G. Cependant, les codes polaires tels que initialement construits pour un décodage à annulations successives (SC) atteignent des performances limitées pour une taille modérée de codes. Cela est lié d'une part à leurs faibles propriétés de distance et d'autre part à la nature du décodage à décision dure. Cependant grâce à l'utilisation d'un décodage par listes principalement ainsi que plusieurs autres améliorations, notamment la pré-transformation, les codes polaires sont désormais compétitifs par rapport aux codes LDPC et aux turbo-codes. Dans ce contexte, cette thèse a pour objet l'étude et l'analyse des codes polaires en se concentrant sur deux aspects fondamentaux qui influencent ces performances : leurs propriétés de distance et leurs performances pour un décodage par listes.

Après une revue approfondie de la définition des codes polaires, des différentes variantes, des algorithmes de décodage et des concepts liés à leur spectre de distances, une première contribution permet de caractériser une partie des propriétés de distance des codes polaires classiques et pré-transformés. Cette méthode présente l'avantage d'être totalement indépendante de la construction code. C'est pourquoi, elle peut être appliquée à différentes configurations. De plus, l'approche proposée se distingue par une complexité de calcul moins élevée que les méthodes présentes dans la littérature. Les techniques de poinçonnage et de raccourcissement des codes polaires sont introduites comme des variantes permettant d'obtenir des codes polaires dont les tailles ne sont pas nécessairement des puissance de deux. Une deuxième contribution consiste à généraliser l'approche développée dans le cadre de la thèse aux codes polaires poinçonnés et raccourcis. Il est à souligner que

cette dernière peut être appliquée quelque soit la technique de poinçonnage et/ou de raccourcissement. Finalement, la question de la taille de liste nécessaire pour un décodage liste (SCL) afin d'atteindre les performances de maximum de vraisemblance est traitée. Celle-ci étant dépendante de la construction du code, un algorithme est proposé afin d'estimer la taille moyenne de liste nécessaire pour atteindre les meilleurs performances de décodage. Cela constitue une contribution très utile pour la construction de codes qui offrent un compromis entre les propriétés de distance et un décodage par liste ayant une complexité calculatoire maîtrisée.

Mots clefs : codes polaires, pré-transformation, spectre de distance, décodage basé sur l'annulation successive à liste, maximum de vraisemblance

Abstract

Error-correcting codes are essential for ensuring reliable data transmission, especially in contexts where various interferences may compromise data integrity. Polar codes are one of the most competitive families of error-correcting codes. They can achieve Shannon channel capacity through efficient encoding and decoding for very large code lengths. For these reasons, polar codes have been included in the 5G standard. Additionally, they are the subject of several research efforts for the future 6G standard. However, polar codes, as originally designed for successive cancellation (SC) decoding, exhibit limited performance for moderate code lengths. This is in part due to their weak distance properties and partly to the nature of hard decision decoding. However, with the use of mainly list decoding and several other enhancements, including pre-transformation, polar codes are now competitive with LDPC and turbo codes. In this context, this thesis aims to study and analyze polar codes focusing on two fundamental aspects that influence their performance: their distance properties and their performance for list decoding.

After a comprehensive review of polar code definition, various variants, decoding algorithms, and concepts related to their distance spectrum, a first contribution characterizes some distance properties of classical and pre-transformed polar codes. This method has the advantage of being entirely independent of code construction, making it applicable to different configurations. Moreover, the proposed approach distinguishes itself by having lower computational complexity than methods in the existing literature.

Polar code puncturing and shortening techniques are introduced as variants to obtain polar codes whose sizes are not necessarily powers of two. A second contribution involves generalizing the developed approach within the thesis to punctured and shortened polar codes. It is noteworthy that this approach can be applied regardless of the puncturing and/or shortening technique used.

Finally, the question of the list size necessary for list decoding (SCL) to achieve maximum likelihood performance is addressed. Since this depends on code construction, an algorithm

is proposed to estimate the average list size required to achieve the best decoding performance. This constitutes a very useful contribution for constructing codes that offer a compromise between distance properties and list decoding with controlled computational complexity.

Key words: polar codes, pre-transformation, distance spectra, successive cancellation list decoding, maximum likelihood

Résumé étendu

Chapitre 1 - Contexte et objectifs

Organisation

Ce chapitre pose le cadre général des communications numériques et introduit les concepts clés qui seront utilisés tout au long de ce manuscrit. Il vise à définir la problématique de la thèse et à exposer ses principaux objectifs. Le chapitre commence par la présentation d'une chaîne de communication numérique classique, décrivant ses éléments fondamentaux : l'émetteur, le canal de transmission, et le récepteur. Le rôle de l'émetteur est de transformer les données en un signal adapté au canal, qui, lui, est souvent affecté par des perturbations telles que le bruit ou les interférences. Le récepteur, quant à lui, est chargé de récupérer les données en corrigeant, si possible, les erreurs introduites par le canal.

Ensuite, la notion de codage de canal est abordée. Cette technique permet d'ajouter de la redondance aux données pour améliorer la fiabilité de la transmission. Le chapitre introduit ensuite les codes polaires [1], qui sont au centre de cette thèse. Ces codes, introduits en 2008 par Erdal Arikan, sont remarquables car ils peuvent, en théorie, atteindre la capacité du canal avec une faible complexité de décodage. Le principe de polarisation, à la base des codes polaires, transforme progressivement un canal en une série de sous-canaux plus ou moins fiables. Ce mécanisme permet aux codes polaires de s'approcher de la limite de capacité en augmentant la taille des blocs de code.

Le décodage par annulation successive (Successive Cancellation, SC), utilisé pour les codes polaires, est ensuite présenté. Cet algorithme décode les bits un à un en utilisant les informations précédemment décodées pour améliorer la précision. Cependant, ses performances sont limitées pour des blocs de petite taille, ce qui est un inconvénient par rapport à d'autres codes correcteurs d'erreurs.

Une section est consacrée aux stratégies de construction des codes polaires, qui reposent sur le choix des bits gelés, ces derniers déterminant la performance du code en fonction de

la fiabilité des sous-canaux. La notion de distance entre les mots de code, qui détermine la capacité à corriger les erreurs, est également abordée. Ce critère est essentiel pour évaluer l'efficacité des codes polaires sous décodage de maximum de vraisemblance. Enfin, le chapitre explore l'impact de la pré-transformation des codes polaires, une technique qui a permis d'améliorer encore davantage leurs performances. Différentes méthodes de pré-transformation sont présentées, ainsi que leurs effets sur les propriétés des codes. Le chapitre se termine par une comparaison des performances des codes polaires selon divers paramètres.

Objectifs de la thèse

Dans le cadre des communications URLLC (Ultra-Reliable Low Latency Communications), le défi majeur consiste à concevoir des systèmes capables de transmettre une grande quantité de données en un temps réduit tout en maintenant une complexité calculatoire faible et une fiabilité élevée. Cela implique d'être en mesure de construire des codes correcteurs d'erreurs, et dans notre cas, des codes polaires, qui offrent un compromis optimal entre performances et complexité. Les performances des codes polaires sont principalement gouvernées par leurs propriétés de distance. Par conséquent, il est essentiel de pouvoir évaluer ces propriétés de distance de manière efficace pour sélectionner les codes polaires les mieux adaptés à des scénarios où des communications ultra-fiables et à faible latence sont requises. Ainsi, il est crucial de trouver des méthodes peu coûteuses en termes de complexité pour déterminer les propriétés de distance des codes polaires. Une telle approche permet non seulement d'améliorer la qualité de la transmission des données, mais aussi d'optimiser les algorithmes de décodage tout en conservant une charge calculatoire raisonnable. La complexité de décodage des codes polaires permettant d'atteindre des performances optimales sont également régies par un certain nombre de paramètres qui seront mis en avant dans ce manuscrit.

Chapitre 2 - Détermination des propriétés de distance des codes polaires

Objectifs

Ce chapitre se concentre sur la proposition d'un algorithme permettant de déterminer la distance minimale ainsi que le nombre d'occurrences associées pour un code polaire, ou plus généralement pour obtenir le spectre réduit d'un code polaire jusqu'à une valeur prédéfinie. Cette approche présente plusieurs avantages notables.

En premier lieu, cet algorithme est déterministe, contrairement à des méthodes probabilistes ou heuristiques. De plus, cet algorithme s'adapte facilement aux codes polaires pré-transformés.

Un autre atout majeur de la méthode proposée est qu'elle ne repose sur aucune hypothèse stricte concernant la construction du code. Cela la rend particulièrement flexible et applicable à toute configuration de bits gelés.

Enfin, l'avantage principal de cet algorithme réside dans sa complexité calculatoire, nettement inférieure à celle des méthodes classiques issues de la littérature. Cet allégement de la complexité calculatoire est essentiel pour des applications nécessitant une analyse rapide et efficace des propriétés de distance.

Principaux résultats

La détermination des propriétés de distance des codes polaires repose principalement sur l'analyse des cosets polaires, qui sont des sous-ensembles spécifiques des codes polaires. Il a été démontré que ces propriétés de distance peuvent être calculées à l'aide d'un algorithme de message-passing sur les graphes de factorisation associés aux codes polaires.

Ce calcul est particulièrement important car il est prouvé que les codes polaires peuvent être décrits comme une union disjointe de cosets polaires spécifiques. Sur cette base, un algorithme est introduit, visant à n'explorer que les cosets qui interviennent dans le calcul des propriétés de distance souhaitées. Cette approche optimise considérablement l'exploration en ne conservant que les éléments significatifs, ce qui réduit la complexité calculatoire globale de manière significative.

En outre, il a été démontré que, de la même manière que pour les codes polaires classiques, les codes polaires pré-transformés peuvent également être exprimés comme une union disjointe de cosets polaires. Cela signifie que l'algorithme proposé est adaptable à ces codes, avec seulement quelques modifications mineures.

De plus, cette méthode a été étendue aux codes polaires concaténés avec un CRC (Cyclic Redundancy Check), ce qui élargit encore son domaine d'application. Le CRC est souvent utilisé pour améliorer la détection et la correction d'erreurs, et l'algorithme est capable de gérer cette concaténation tout en maintenant une faible complexité.

Les résultats expérimentaux obtenus pour la détermination des spectres de distance sont conformes aux résultats déjà présents dans la littérature, ce qui confirme la validité et la précision de la méthode proposée. En outre, une comparaison de la complexité calculatoire

de cette approche par rapport aux méthodes existantes montre que l'algorithme proposé permet de réduire cette complexité de plusieurs ordres de grandeur, rendant la méthode plus efficace.

Chapitre 3 - Détermination des propriétés de distance des codes polaires poinçonnés et raccourcis

Objectifs

Ce chapitre s'intéresse à la généralisation des principes abordés dans le chapitre afin de calculer les propriétés de distance des codes polaires poinçonnés et raccourics.

Habituellement, les codes polaires sont construits avec des tailles qui sont des puissances de 2. Cela limite leur applicabilité dans les scénarios nécessitant des longueurs de bloc courtes. Pour surmonter cette limitation, des techniques telles que le poiçonnage et le raccourcissement ont été introduites. Ces méthodes permettent de construire des codes polaires avec des longueurs de code plus flexibles.

La détermination des propriétés de distance des codes polaires poinçonnés et raccourcis est cruciale pour optimiser leurs performances sous un décodage à maximum de vraisemblance. Cette connaissance permet d'optimiser les schémas de poiçonnage et de raccourcissement, ainsi que la construction de ces codes, afin d'atteindre les objectifs de performance souhaités. De plus, il est essentiel de prendre en compte l'impact de la pré-transformation sur les codes polaires poinçonnés et raccourcis pour améliorer leurs propriétés de distance.

Pricipaux résultats

La détermination des propriétés de distance pour les codes polaires poinçonnés et raccourcis repose essentiellement sur l'introduction de cosets polaires également soumis aux opérations de poinçonnage et de raccourcissement. L'avantage de cette approche réside dans la prise en compte directe de l'impact de ces opérations au sein même des cosets polaires. Des adaptations spécifiques sont ensuite mises en place en fonction des motifs de poinçonnage et de raccourcissement appliqués.

Il est par la suite démontré que, tout comme les codes polaires classiques, les codes polaires poinçonnés ou raccourcis peuvent également être exprimés comme une union disjointe de cosets polaires poinçonnés ou raccourcis spécifiques. Ce résultat permet d'appliquer une stratégie similaire à celle utilisée pour les codes polaires classiques, en

effectuant un élagage des cosets afin de ne conserver que ceux qui interviennent dans le calcul des propriétés de distance recherchées. Cette étape permet de réduire la complexité des calculs tout en maintenant l'aspect déterministe de la méthode.

Cette méthode est ensuite adaptée aux codes polaires poinçonnés et raccourcis avec pré-transformation, en imposant une contrainte spécifique sur le motif de raccourcissement dans le cas des codes polaires raccourcis.

Les résultats expérimentaux obtenus confirment la validité de cette approche. En effet, les résultats sont alignés avec ceux présents dans la littérature, tout en offrant une réduction significative de la complexité calculatoire par rapport aux méthodes existantes. Cette réduction de la complexité rend la méthode proposée plus adaptée aux applications pratiques.

Chapitre 4 - Estimation de la taille de liste moyenne permettant d'atteindre les performances de maximum de vraisemblance

Objectifs

Ce chapitre introduit une méthode pour optimiser la construction de codes polaires, en se concentrant sur leurs propriétés de distance. Les codes polaires conçus uniquement en fonction de ces contraintes montrent des performances moindres lors du décodage par liste avec annulation successive (SCL), surtout avec une taille de liste modérée. Nous explorons ici les paramètres influençant la taille moyenne de la liste nécessaire pour obtenir des performances proches de celles du maximum de vraisemblancce, tout en limitant la complexité calculatoire.

De plus, il est expliqué comment certains chemins de décodage peuvent être éliminés lors du décodage par liste sans affecter les performances globales, ce qui permet de réduire la complexité tout en maintenant un haut niveau de fiabilité.

Principaux résultats

Dans ce chapitre, une méthode de construction de codes polaires basée uniquement sur leurs propriétés de distance est présentée. Il est démontré que, pour des codes de taille modérée, cette méthode permet d'atteindre des performances proches de celles du Maximum de Vraisemblance (ML), lorsqu'un décodage par liste avec une taille de liste

modérée est utilisé. Cependant, pour des codes de plus grande taille, les performances sous un décodeur par liste à taille modérée se dégradent significativement, rendant cette approche moins efficace.

La deuxième partie du chapitre se concentre donc sur l'analyse de l'impact de la construction du code sur la taille de la liste nécessaire pour s'approcher des performances du décodage ML. Il est d'abord démontré que le décodage par liste des codes polaires peut être simplifié à partir d'une certaine étape sans compromettre les performances. Ensuite, il est prouvé que la taille moyenne de la liste permettant d'atteindre les performances de ML est étroitement liée aux propriétés de distance de certains cosets polaires spécifiques tout au long du processus de décodage.

L'exploitation des outils développés dans le chapitre 2 ainsi que la définition précise des facteurs générant le besoin en taille de liste pour les codes polaires permettent d'estimer l'évolution de ce besoin en taille de liste tout au long du décodage.

Les résultats expérimentaux confirment que la méthode proposée offre une estimation précise du besoin en taille de liste pour atteindre les performances de décodage ML. De plus, il est montré comment cette méthode, combinée avec celles développées dans les chapitres 2 et 3, permet de construire des codes optimisant le compromis entre performances et complexité calculatoire lors du décodage.

Table of contents

List of figures

List of tables

List of acronyms

5*G*	Fifth generation of cellular mobile communications
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
B-DMC	Binary Discrete Memoryless Channel
BPSK	Binary Phase Shift Keying
CA-SCL	CRC Aided Successive Cancellation List
CDF	Cumulative Distribution Function
CRC	Cyclic Redundacy Check
DE	Density Evolution
DE-GA	Density Evolution by Gaussian Approximation
DFB	Dynamic Frozen Bits
DMC	Decreasing Monomial Code
DTPM	Difference to True Path Metric
FER	Frame Error Rate
LLR	Log Likelihood Ratio
MC	Meta Converse
ML	Maximum Likelihood
MWEF	Minimum Weight Enumeration Function
PAC	Polarization-adjusted convolutional
PDF	Probability Distribution Function
RCU	Random Coding Union
RM	Reed-Muller
RWEF	Reduced Weight Enumeration Function
SC	Successive Cancellation
SCL	Successive Cancellation List
SNR	Signal-to-noise Ratio
TUB	Truncated Union Bound
UB	Union Bound
URLLC	Ultra-Reliable Low Latency Communications

WEF	Weight Enumeration Function

Introduction

In a typical digital communication setup, a source generates information, a channel serves as the medium for transmitting data, and a receiver at the receiving end. In order to ensure reliable transmission of information from the source to the receiver, it is crucial to address the various distortions introduced by the channel due to multiple factors.

To address this challenge, channel coding is integrated into digital communication chains. This technique is employed in communication systems to enhance the reliability of data transmission. By adding redundancy to the transmitted data, error control coding enables the detection and, in some cases, the correction of errors that may arise during transmission. This approach plays a major role in mitigating the effects of channel impairments and ensuring the fidelity of transmitted data in digital communication systems.

In his seminal work "The Mathematical Theory of Communication" [2], Claude Shannon revolutionized the understanding of information transmission. Among his groundbreaking contributions was the formulation of the channel capacity, a fundamental concept in information theory. Shannon also demonstrated the significance of encoding in his second theorem, illustrating how channel coding can introduce redundancy to the original transmission. It enables to facilitate dependable information transfer at rates below the channel's capacity. Despite Shannon's groundbreaking theoretical insights, explicit codes capable of achieving his theoretical limit have yet to be determined. Since the demonstration of his mathematical framework, extensive research has been devoted to developing codes that can approach this limit. Beginning with simple repetition codes, which involve transmitting information multiple times to infer the received message, researchers have progressed to more sophisticated error-correcting codes such as BCH [3], Reed-Solomon [4], convolutional codes [5], turbo codes [6] and LDPC codes [7]. The main objective is to get closer to the theoretical limits while maintaining reasonable computational complexity adapted to specific use cases. Despite these advancements, reaching Shannon limit remains a significant challenge.

Arikan's introduction of polar codes in 2008 [1] is considered as a new step by offering a method capable of asymptotically achieving channel capacity with a low-complexity decoder, based on the Successive Cancellation algorithm. Besides, they accommodate variable code rates and lengths, with straightforward mechanisms in place for puncturing and shortening. However, these advantages come with a trade-off: to fully harness their potential, polar codes require transmission of very large messages. For shorter to moderate block lengths, polar codes have demonstrated mediocre performance. This makes them less appealing compared to established error-correcting codes such as turbo codes and LDPC codes.

Over the past decade, significant efforts have been directed towards enhancing the error correction capabilities of polar codes. These efforts have primarily focused on two aspects: refining practical decoding algorithms and enhancing the code construction process. Various proposed decoding algorithms, among them the Successive Cancellation List decoder [8], have proven to be instrumental in achieving competitive error correction performance, particularly for short block-lengths. Thanks to these improvements, polar codes have emerged in the field of error correction codes, underscored by their inclusion in the fifth generation of mobile telephony standards 5G [9]. They also are subject of extensive research for the future 6G standard.

Interest in the construction of polar codes adapted to decoding strategies has only recently gained momentum, particularly with the emergence of several pre-transformation techniques [10–12]. These techniques have demonstrated superior distance properties compared to traditional polar codes, sparking renewed attention in optimizing code construction for enhanced performance.

This thesis focuses on two critical parameters that dictate the performance of polar codes: their distance properties and their ability to achieve Maximum Likelihood performance when decoded with successive list algorithm. These aspects serve as essential factors in the design of codes and a lower decoding complexity. The investigation extends to pre-transformed polar codes and variants that have undergone shortening and puncturing. These techniques are particularly significant as they allow for the definition of codes with block lengths that may not necessarily conform to powers of 2, thereby broadening the scope of achievable code lengths. The dissertation is organized into four chapters, each focusing on a different aspect of the research studies. **In the first chapter**, the focus is on understanding polar codes and the fundamental polarization phenomenon that defines the way they operate, allowing them to approach the channel's capacity under successive cancellation (SC) decoding. The chapter also delves into conventional code construction methods tailored for SC decoding. A brief overview of the distance

properties exhibited by polar codes is provided. Various pre-transformation techniques for polar codes are introduced, expanding the available techniques for code enhancement. Additionally, alternative decoding methods beyond the successive cancellation decoder are discussed. They offer insights into the trade-off between decoding performance and computational complexity.

In the second chapter, a first contribution of this manuscript is introduced: the computation of the distance properties of pure and pre-transformed polar codes. The weight distribution of error-correcting codes provides insights into their performance under Maximum Likelihood decoding. Therefore, identifying the most significant elements of the weight spectrum is crucial for characterizing their performance. The method proposed relies on computing the distance properties of *polar cosets* and subsequently pruning the irrelevant ones. Notably, one of its primary advantages is its independence from constraints imposed on code construction or the pre-transformation techniques applied. This approach offers a low-complexity computation, compared to existing methods of assessing the distance properties of polar codes, thereby facilitating design decisions.

In the third chapter, the insights gained from the second chapter are extended to generalize the proposed method to punctured and shortened pure and pre-transformed polar codes. This method also benefits from the fact of being totally independent of the chosen puncturing or shortening scheme. Traditionally, polar codes are constrained to block lengths expressed as powers of 2. However, puncturing and shortening techniques offer a departure from this rule, providing increased flexibility to polar codes. Recent studies, such as [13], have demonstrated that, similarly to classic polar codes, punctured and shortened polar codes under specific schemes can achieve channel capacity. Given the growing interest in ultra-reliable low latency communications (URLLC), where moderate block lengths are prevalent, the exploration of punctured and shortened polar codes becomes increasingly relevant. Understanding the distance properties of such codes is essential to design code tailored to these specific scenarios.

The fourth chapter focuses on investigating the list size requirements when employing a Successive Cancellation List decoder to achieve performances comparable to Maximum Likelihood decoding. While the distance properties of polar codes define these performances, the computational complexity associated with decoding can be prohibitively high, especially for moderate to large block lengths. This chapter delves into various parameters influencing the evolution of the list size necessary to achieve best code performance. Specifically, it is demonstrated that, similarly to distance properties, the list size required to achieve Maximum Likelihood performance is heavily influenced by the characteristics of pertinent polar cosets. This approach offers a significant advantage of

adaptability to pre-transformation, shortening, and puncturing techniques, making it highly attractive for code construction purposes.

Finally, this thesis concludes with a recapitulation of the various contributions put forth throughout the manuscript. Additionally, the thesis outlines several perspectives for future exploration and development based on the insights gained from the conducted research. These perspectives shed light on potential areas for further investigation and innovation in the short, medium and long terms.

List of contributions

The main contributions of this thesis can be summarized as follows:

1. A low-complexity algorithm for determining the distance properties of polar and precoded polar codes. This algorithm is "universal" in the sense that it is independent of the code construction and precoding, allowing it to adapt to any scenario.
2. An extension of the initial algorithm to compute the distance properties of punctured and shortened precoded polar codes. Experimental results demonstrate that the proposed computation significantly reduces computational complexity by several orders of magnitude compared to existing methods.
3. An algorithm for constructing precoded polar codes based on their distance properties.
4. An upper bound on the maximum list size necessary to achieve maximum likelihood decoding performance. This bound can also be applied to punctured and shortened polar codes.

These various contributions have been highlighted through scientific publications:

- **National conferences** :
  - M. Ellouze, C. Leroux, R. Tajan, C. Poulliat et C. Jégo, Décodage SC par listes optimisées de codes polaires. in *GRETSI*, 2022.
  - M. Ellouze, R. Tajan, C. Leroux, C. Jégo et C. Poulliat, Algorithme faible complexité pour le calcul de la distribution de la distance minimale de codes polaires. in *GRETSI*, 2023.

- **International conferences** :
  - M. Ellouze, C. Leroux, R. Tajan, C. Poulliat et C. Jégo, Tailored List Decoding of Polar Codes. in *11th International Symposium on Topics in Coding (ISTC)*, 2021.
  - M. Ellouze, R. Tajan, C. Leroux, C. Jégo et C. Poulliat, Low-complexity algorithm for the minimum distance properties of PAC codes. in *12th International Symposium on Topics in Coding (ISTC)*, 2023.
- **International journals** :
  - M. Ellouze, R. Tajan, C. Leroux, C. Jégo et C. Poulliat, Enumeration of Low-Weight codewords of Punctured and Shortened Pre-Transformed Polar Codes. to be submitted in *IEEE Transactions on Communications*.

1 Generalities on Polar codes

This chapter provides an overview of key concepts in channel coding, focusing on polar coding, to lay the groundwork for subsequent chapters. Specifically, it covers topics such as channel polarization, techniques for constructing polar and pre-transformed polar codes, properties of these codes, and associated decoding strategies.

1.1 Principle of polar codes

1.1.1 Channel coding

While digital communication chains can present some differences, they share a foundational model established by Claude Shannon in [2]. This model describes the key steps in transmitting an initial message $\mathbf{u}$ from a source point to a receiver through a transmission channel, which serves as the transmission medium for the information. The objective is thus to achieve flawless recovery of the transmitted signal upon reception. When a signal is transmitted through a communication channel, it undergoes numerous perturbations primarily stemming from thermal noise generated by electronic components within the transmission chain. In the case where the aforementioned perturbations are of high significance, they have the potential to distort the originally transmitted message, thereby introducing errors into the received message. To tackle this issue, the communication chain model underwent refinement by incorporating a channel encoder and decoder, a process known as channel coding. The primary objective of channel coding is to enable data transmission that is both more reliable and efficient. The encoding consists in transforming an information sequence of K bits into a codewords of N bits where $N > K$. This means that redundancy is added to the initially transmitted message, i.e., the code rate $R = \frac{K}{N} < 1$. The model of a coded communication chain is given in Figure 1.1. In this model, the source generates a message $\mathbf{u}$, which is then encoded into

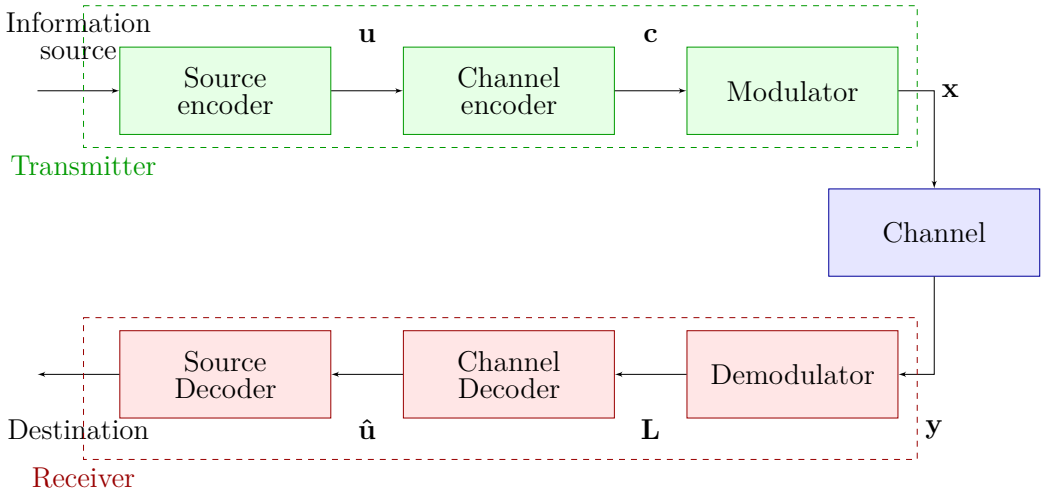


Figure 1.1: Digital communication chain

a codeword $\mathbf{c}$ and modulated into a vector of symbols $\mathbf{x}$ for transmission through the channel. The altered message $\mathbf{y}$ upon passage through the channel undergoes the reverse operations of the transmitter side to obtain an estimated version of the transmitted message $\hat{\mathbf{u}}$. The aim is therefore to correctly recover the transmitted message, i.e., $\mathbf{u} = \hat{\mathbf{u}}$. Among Shannon's results [2], it is asserted that if the code rate is below a threshold determined by the channel capacity defined as the maximal amount of information that a channel can transport, there exist encoding and decoding schemes such that the originally transmitted message can be reliably recovered asymptotically. The main open question revolves around discovering encoding and decoding techniques that offer reliability while maintaining low complexity. Polar codes [1] are the first deterministic family of error correcting codes to achieve the channel capacity asymptotically for a certain class of channels under a low complexity decoding algorithm called Successive Cancellation (SC) decoder that will be detailed in the next sections.

1.1.2 Binary-Discrete Memoryless Channels (B-DMC)

Let us consider W, a channel. W is considered as a Binary-Discrete Memoryless Channel (B-DMC) if it obeys to the following properties:

- The input alphabet $\mathcal{X}$ is discrete.
- The output alphabet $\mathcal{Y}$ is discrete.
- For $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, the conditional probability is denoted by $W(y|x)$. Due to the memoryless property of the channel, and given $\mathbf{x} = (x_0, x_1, ..., x_{N-1})$ and $\mathbf{y} = (y_0, y_1, ..., y_{N-1})$, the following property is verified:

$$W_N(\mathbf{y}|\mathbf{x}) = \prod_{i=0}^{N-1} W(y_i|x_i) \tag{1.1}$$

The symmetric capacity $I(W)$ of a B-DMC channel is given by [1]:

$$I(W) = \sum_{\substack{x \in \mathcal{X} \\ y \in \mathcal{Y}}} \frac{1}{2} W(y|x) \log \frac{2W(y|x)}{W(y|0) + W(y|1)} \tag{1.2}$$

This symmetric capacity represents a rate measure that is helpful to determine the maximum rate at which a reliable communication can be achieved across the channel W while using uniform inputs.

1.1.3 Polar codes

Let $G_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ denote the *kernel* matrix and $\mathbf{u} = [u_0, u_1]$ an input sequence. The encoded sequence is $\mathbf{x} = [x_0, x_1] = [u_0 \oplus u_1, u_1]$. Let us define the B-DMC channel W_2 where:

- The input vector is $[u_0, u_1] \in \mathcal{X}^2$.
- The output vector is $[y_0, y_1] \in \mathcal{Y}^2$
- The transition probability

$$W_2(\mathbf{y}|\mathbf{u}) \triangleq W_2(\mathbf{y}|\mathbf{u}\mathbf{G}_2) = W(y_0|u_0 + u_1)W(y_1|u_1) \tag{1.3}$$

This channel can be split into two artificial channels as presented in [1]:

- A first channel $W_2^{(-)} : \mathcal{X} \to \mathcal{Y}^2$ for which the input is u_0 and the output is $[y_0, y_1]$.
- A channel $W_2^{(+)} : \mathcal{X} \to \mathcal{Y}^2 \times \mathcal{X}$ for which the input is u_1 and the output is $[y_0, y_1, u_0]$.

Using probability rules, $W_2^{(-)}$ and $W_2^{(+)}$ can be computed as:

$$W_2^{(-)}(y_0, y_1|u_0) = \frac{1}{2} \sum_{u_1 \in \mathcal{X}} W_2(y_0, y_1|u_0, u_1) = \frac{1}{2} \sum_{u_1 \in \mathcal{X}} W(y_0|u_0 \oplus u_1)W(y_1|u_1) \tag{1.4}$$

$$W_2^{(+)}(y_0, y_1|u_0) = \frac{1}{2} W_2(y_0, y_1|u_0, u_1) = \frac{1}{2} W(y_0|u_0 \oplus u_1)W(y_1|u_1) \tag{1.5}$$

Channels $W_2^{(-)}$ and $W_2^{(+)}$ can be interpreted as follows. As u_0 has an influence on the values of both y_0 and y_1, this offers comparatively more information about u_1 than about u_0. Moreover, because of the transmission of $u_0 \oplus u_1$, information of u_0 and u_1 are combined, leading to less information about u_0. It has been shown in [1] that:

$$I(W_2^{(-)}) \leq I(W) \leq I(W_2^{(+)}) \tag{1.6}$$

This means that starting from a B-DMC channel W, one synthetic more reliable channel $W_2^{(+)}$ and another less reliable synthetic channel $W_2^{(-)}$ can be obtained.
The construction of polar codes deeply relies on the extension of the transformation of a set of channels (W, W) to a synthetic set of channels $(W_2^{(-)}, W_2^{(+)})$ of lower and higher

reliability. The construction of polar codes highly relies on this channel transformation or *polarization* that can be extended to a high number of channels.
A polar code $\mathcal{C}(N, K)$ is a linear code block with a size $N = 2^n$. The transformation matrix of a polar code is given by the n-fold polar Kronecker matrix $\mathbf{G}_N = \mathbf{G}_2^{\bigotimes n}$ where $\mathbf{G}_2 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ and $(\cdot)^{\bigotimes n}$ denotes the n^{th} Kronecker product power. Figure 1.2 illustrates different generator matrices $\mathbf{G}_N$ as well as their associated encoding schemes.

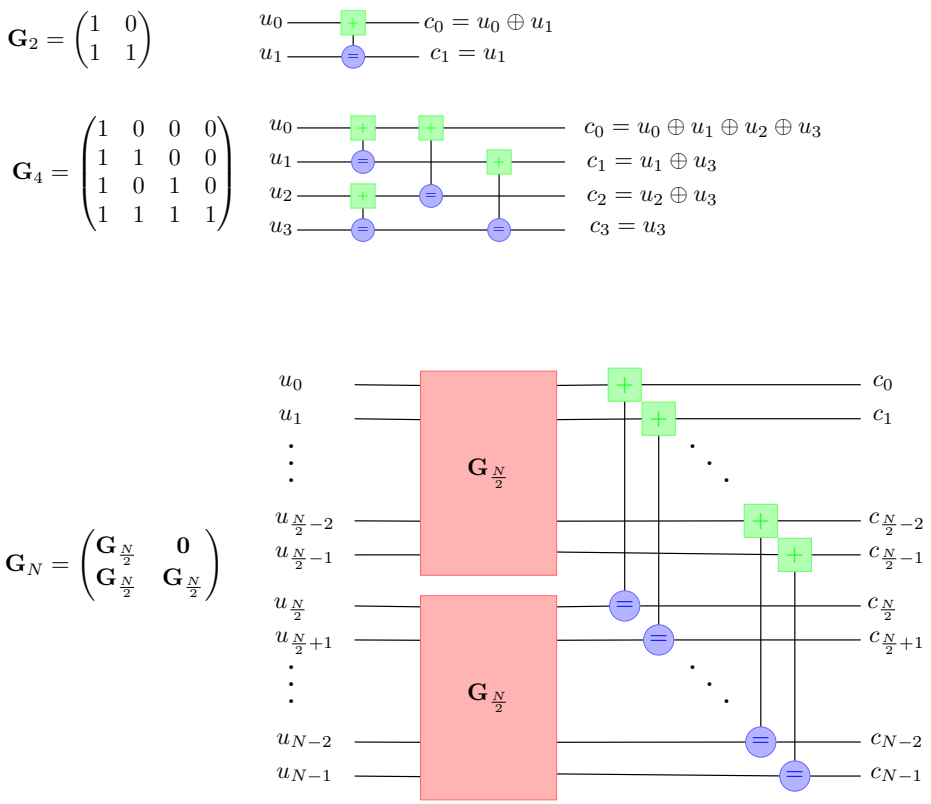


Figure 1.2: Different polar codes and the corresponding encoding schemes

The recursive construction can be observed in Figure 1.2. It stems from the fact that $\mathbf{G}_{\frac{N}{2}} \bigotimes \mathbf{G}_2 = \mathbf{G}_N$. Therefore, starting from N independent channels W, it is possible to build N synthetic channels $W_N^{(i)} : \mathcal{X} \to \mathcal{Y}^N \times \mathcal{X}^{i-1}$ where:

$$W_N^{(i)}(\mathbf{y}, u_0^{i-1}|u_i) = \sum_{u_{i+1}^N \in \{0,1\}^{i-1}} \frac{1}{2^{N-i}} W_N(\mathbf{y}|\mathbf{u}\mathbf{G}_N). \tag{1.7}$$

Arikan proved in [1] that asymptotically in N, the capacities $I(W_N^{(i)}(\mathbf{y}, u_1^{i-1}|u_i))$ either tend to 0 or to 1. This means that the synthetic channels are divided into two groups. A

group of very reliable channels whose symmetric capacity tends to 1 where the information if perfectly transmitted and a group of very unreliable channels with a channel capacity of 0.
Defining a polar code $\mathcal{C}(N, K)$ therefore boils down to pinpointing the K positions where the information bits are transmitted, whereas the remaining ones are *frozen* (i.e. they are set to a known value). In all the following, we denote by *pure polar codes* [14] polar codes with all the frozen bits equal to 0. The set of frozen bits is denoted by $\mathcal{F}$ and the information bit set by $\mathcal{I}$. We will discuss in further sections the different ways to define the information and frozen sets. Figure 1.3 represents the encoding process for polar codes. Starting from a bit vector **b** of length K, the vector **u** is obtained from **b** by inserting the information bits of **b** in **u**. The remaining positions of **u** are frozen. This operation is called *rate-profiling*.

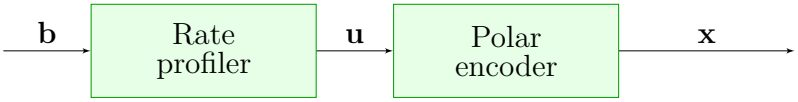


Figure 1.3: Polar encoding

1.1.4 Successive Cancellation decoding of polar codes

The decoding of polar codes using the Successive Cancellation (SC) highly relies on the encoding construction. In fact, as polar codes construction can be illustrated by factor graphs, the encoding consists in the propagation of the information from the left side of the graph towards the right side of the graph. By contrast, the decoding consists in the propagation of the channel observation on the right of the graph representing the different bit estimations. Those estimations are expressed as Log Likelihood Ratio (LLR) L_i as:

$$L_i = \log\left(\frac{P(u_i = 0|y_i)}{P(u_i = 1|y_i)}\right) \tag{1.8}$$

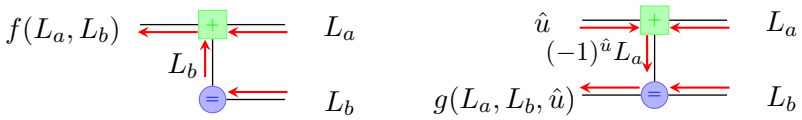
The SC decoding process estimates each u_i from the received word y_1^N and the past decisions $\hat{u}_1^{i-1}$ as:

$$\hat{u}_i = \begin{cases} 0 \text{ if } L_i \geq 0 \quad \text{or} \quad \text{if} \quad i \in \mathcal{F} \\ 1 \text{ otherwise} \end{cases} \tag{1.9}$$

where L_i is also defined as:

$$L_i = \log\left(\frac{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|u_i = 0)}{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|u_i = 1)}\right) \tag{1.10}$$

The computation of L_i is done using the different propagation rules in equations (22) and (23) of [1]. During decoding, two distinct configurations may arise based on the encountered node. The first configuration represented in Figure 1.4 refers to the f function case whereas the second configuration refers to the g function case.



(a) f function case (b) g function case

Figure 1.4: Graph factor configurations for LLR computation

The different configurations correspond to the following functions [15]:

$$\begin{cases} f(L_a, L_b) = \text{atanh}(\tanh(\frac{L_a}{2})\tanh(\frac{L_b}{2})) \\ g(L_a, L_b, \hat{u}_0) = (-1)^{\hat{u}_0} L_a + L_b \\ h(\hat{u}_0, \hat{u}_1) = (\hat{u}_0 \oplus \hat{u}_1, \hat{u}_1) \end{cases} \tag{1.11}$$

Note that $f(L_a, L_b)$ can also be approximated by:

$$f(L_a, L_b) \approx \text{sign}(L_a L_b)\min(L_a, L_b) \tag{1.12}$$

The overall computations of an SC decoder are summarized in Figure 1.5. In this figure, we denote by $\mathbf{L}^{(i)} = [L_0^{(i)}, L_1^{(i)}, \cdots, L_{N-1}^{(i)}]$ the LLR values at the depth i of the decoding tree. Algorithms 1 and 2 describe the overall operations for computing the LLR associated to the i^{th} synthetic channel and the overall SC decoding process respectively.

1.2 Construction and properties of polar codes

The initial stage in defining a polar code, known as code construction or rate-profiling, involves identifying the most reliable virtual channels for transmitting information. The primary methods employed for this purpose will be discussed in the next sections. It is essential to note beforehand that the reliability of virtual channels primarily hinges on channel characteristics.

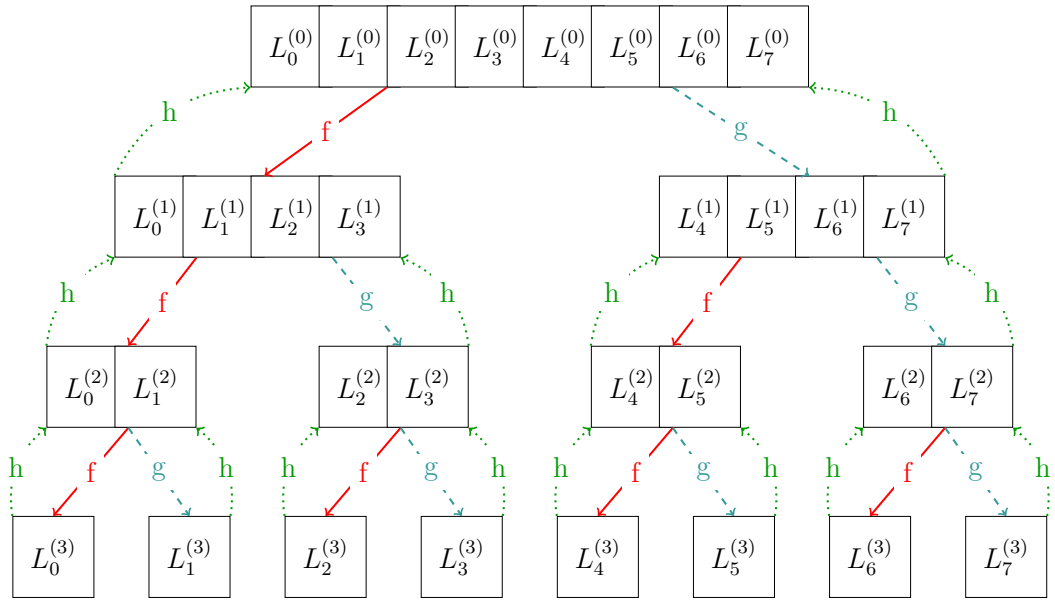


Figure 1.5: Successive Cancellation decoding of a $N = 8$ polar code

Algorithm 1: ComputeLLR(N, $\boldsymbol{\lambda}$, $\hat{\mathbf{u}}_0^{i-1}$)

Input: Code length N, channel LLRs $\boldsymbol{\lambda}$ and previously estimated $\hat{u}_0^{i-1}$
Output: LLR associated to the i^{th} synthetic channel L_i

if $N = 1$ **then**
  return λ_0
end
else
  if $i \mod 2 = 0$ **then**
    $l_1 \leftarrow \text{ComputeLLR}(\frac{N}{2}, \boldsymbol{\lambda}_0^{\frac{N}{2}-1}), \mathbf{u}_{0,even}^{i-1} \oplus \mathbf{u}_{0,odd}^{i-1})$
    $l_2 \leftarrow \text{ComputeLLR}(\frac{N}{2}, \boldsymbol{\lambda}_{\frac{N}{2}}^{N-1}, \mathbf{u}_{0,odd}^{i-1})$
    return $f(l_1, l_2)$ `/* f function */`
  end
  else
    $l_1 \leftarrow \text{ComputeLLR}(\frac{N}{2}, \boldsymbol{\lambda}_0^{\frac{N}{2}-1}), \mathbf{u}_{0,even}^{i-2} \oplus \mathbf{u}_{0,odd}^{i-2})$
    $l_2 \leftarrow \text{ComputeLLR}(\frac{N}{2}, \boldsymbol{\lambda}_{\frac{N}{2}}^{N-1}, \mathbf{u}_{0,odd}^{i-2})$
    return $g(l_1, l_2, \hat{u}_{i-1})$ `/* g function */`
  end
end

Algorithm 2: SCDecode$(N, \boldsymbol{\lambda}, \mathcal{F})$

Input: Code length N, Channel LLRs $\boldsymbol{\lambda}$, frozen set $\mathcal{F}$
Output: Bit vector estimation $\hat{\mathbf{u}}$

```
for i ← 0 to N − 1 do
    if i ∈ F then
        L_i ← ComputeLLR(N, λ, û_0^{i−1})
        û_i ← 0
    end
    else
        L_i ← ComputeLLR(N, λ, û_0^{i−1})
        û_i ← (1/2)(1 − sign(L_i))
    end
    return û_0^{N−1}
end
```

1.2.1 Partial order of synthetic channels

In this section, we introduce the partial order of synthetic channels, which asserts that certain channels offer superior reliability under SC decoding compared to others. This partial order constraint is used to construct rate-profiling strategies for polar codes under SC decoding. It is a topic we discuss in Section 1.2.2.
In Section 1.2.3, we explore how the partial order facilitates straightforward methods for determining certain distance properties of polar codes.

It has been shown in [16] that polar codes can be interpreted as monomial codes. Under this assumption, every synthetic channel $W_N^{(i)}(\mathbf{y}, \mathbf{u}_0^{i-1}|u_i)$ can be formulated as a monomial. For all $j \in [0, N-1]$, we denote with $\mathbf{b}^{(j)} = (b_0^j, b_1^j, \cdots, b_{n-1}^j)$ the binary representation of j associated to the j^{th} row of the matrix $\mathbf{G}_N$ with b_0^j being the most significant bit. For each row $\mathbf{G}_N^{(j)}$ we associate a monomial $f_j(m_0, \cdots, m_{n-1})$ that is expressed as:

$$f_j(m_0, \cdots, m_{n-1}) = \prod_{i \in [0, n-1]} m_i^{1-b_i^j} \tag{1.13}$$

Example 1.2.1. *Let us consider the polar code with $N = 8$. Figure 1.6 describes the monomials associated to the rows of the matrix $\mathbf{G}_8$. As an example, the binary representation of the fourth row is $(0, 1, 1)$, the associated monomial is therefore $f_3 = m_0$*

It has been demonstrated in [16] and [17] that synthetic channels obey to a partial order

$$\begin{array}{cc} m_0m_1m_2 & u_0 \\ m_0m_1 & u_1 \\ m_0m_2 & u_2 \\ m_0 & u_3 \\ m_2m_2 & u_4 \\ m_1 & u_5 \\ m_2 & u_6 \\ 1 & u_7 \end{array} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Figure 1.6: Monomials for a polar code with $N = 8$

denoted by the operator $\preceq$ based on their reliabilities. The relation $f_j \preceq f_k$ is verified if the synthetic channel $W_N^{(j)}(\mathbf{y}, u_0^{j-1}|u_j)$ is more reliable than the synthetic channel $W_N^{(k)}(\mathbf{y}, u_0^{k-1}|u_k)$. For every monomial $f = \prod_{j=0}^{d-1} m_{i_j}$, we assume that $i_0 < i_1 < \cdots < i_{d-1}$ and we denote by $\deg(f) = d$ and $\deg(g) = l$. For two monomials $f = \prod_{j=0}^{d-1} m_{i_j}$ and $g = \prod_{k=0}^{l-1} m_{i_k}$, the condition $f \preceq g$ is satisfied under the following conditions [16]:

- if $\deg(f) = \deg(g)$, i.e., $d = l$, then:

$$f \preceq g \Leftrightarrow i_k \leq j_k \forall \quad k \in [0, d-1]$$

- if $\deg(f) < \deg(g)$, the following rule is applied:

$f \preceq g$ if there exists g^* a divisor of g such that $\deg(g^*) = \deg(f)$ and $f \preceq g^*$

For polar codes under SC decoding that reaches Shannon's capacity, the partial order has to be taken into account during the construction of the rate-profiling. The rate-profiling methods presented in Section 1.2.2 is consistent with the partial order criterion. Note that those rate-profile constructions are only optimal under SC decoding. We will discuss in further sections the efficiency of those constructions under different decoding processes and for pre-transformed polar codes.

1.2.2 Rate-profiling construction

In this section, we present the existing mostly used rate-profiling for polar codes. The first rate-profiling construction was introduced by Arikan in [1] and relies on a heuristic determination. This method relies on Monte Carlo simulation. Therefore it depends on the channel realisation. Other approaches [18, 19] use a Density Evolution (DE) computation based on message propagation similar to the structure of successive cancellation decoding.

Other construction methods such as Reed-Muller (RM) [20] rate profiling rely on the line weight properties of the generator matrix. Further rate profiling construction will be discussed in Chapter 4. It is important to highlight that the rate-profiling highly impact the distance properties of polar codes. This will be further discussed in Section 1.2.3.

1.2.2.1 Density Evolution rate-profiling

The Density Evolution method essentially relies on using the same SC decoding factor graphs to propagate the Probability Density Function (PDF) of the LLRs. For an Additive White Gaussian Noise (AWGN) channel, this construction is streamlined and referred to as DE-GA (Density Evolution using Gaussian Approximation). As the channel LLRs can be expressed as $L_i = \frac{2y_i}{\sigma^2}$ and under the Gaussian assumption, and admitting that the all- zero codeword is transmitted, $L_i \sim \mathcal{N}(\frac{2}{\sigma^2}, \frac{4}{\sigma^2})$. It is supposed at every decoding stage that L_i is Gaussian and that its mean and variance obey to $\mathbb{V}(L_i = 2\mathbb{E}(L_i)$. Hence, typically, only the mean value is computed. Indeed, there exists a straightforward method to infer the variance. Similarly to the SC decoding process, the mean values of $L_i^{(d)}$ are propagated through factor graphs using the rules described in Equations (5) and (6) of [19]. The DE-GA method is widely utilized in practice because it deviates from the classic DE method only for codes with very large sizes (around $N = 2^{18}$) [21].

1.2.2.2 Reed-Muller construction

The Reed-Muller (RM) code [20] denoted by $\mathcal{RM}(r, m)$ where $0 \leq r \leq m$ is obtained by selecting the indexes associated to the rows of the matrix $\mathbf{G}_{2^m}$ having a Hamming weight greater or equal to 2^{m-r}.

Example 1.2.2. *Let us consider the RM code $\mathcal{RM}(1, 3)$. In this case:*

$$\mathbf{G}_8 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \tag{1.14}$$

The generator matrix of $\mathcal{RM}(1, 3)$ is obtained by selecting the rows of $\mathbf{G}_8$ having a weight greater or equal to 4 (rows highlighted in blue). Therefore, it comes down to freeze the

first, second, third and fifth line, i.e. $\mathcal{F} = \{0, 1, 2, 4\}$.

1.2.2.3 RM-polar constructions

The RM-polar [22] rate-profiling constructions rely on combining both RM and polar code constructions. We can see from Section 1.2.2.2 that the RM construction cannot be applied to any desired rate. Let $\mathcal{RM}(r, m)$ represent an RM code, and let k denote the number of rows with a weight less than or equal to 2^{m-r}. The RM-polar construction operates as follows in order to build a $\mathcal{C}(N, K)$ polar code:

- Select r such that $k \leq K$
- Initialise the information set $\mathcal{I}$ as $\mathcal{I} = \left\{ i \in [\![0; N-1]\!] | w(\mathbf{G}_N^{(i)}) \geq 2^{m-r}), \right\}$, where $\mathbf{G}_N^{(i)}$ denotes the i^{th} line of G_N and $w(\mathbf{c})$ defines the Hamming weight of a codeword $\mathbf{c} \in \mathbb{F}_2^N$.
- Choose the remaining $K - k$ information bits with the highest reliability using one of the previous methods (e.g; density evolution)

We will show in Section 1.2.3 that the RM-polar constructions yields better distance properties for polar codes.

1.2.3 Distance properties of polar codes

We discuss in this section the impact of the rate-profiling on the distance properties of polar codes, i.e. the hamming weights of the generated codewords. The Hamming distance $d(\mathbf{c}, \check{\mathbf{c}}) = w(\mathbf{c} \oplus \check{\mathbf{c}})$ between two codewords $\mathbf{c}$ and $\check{\mathbf{c}}$ is defined as the number of components for which they differ. The minimum distance of the linear code $\mathcal{C}$ is given by

$$d^*(\mathcal{C}) = \min_{\mathbf{c} \in \mathcal{C}, \mathbf{c} \neq \mathbf{0}} w(\mathbf{c}) \tag{1.15}$$

Understanding the distance spectrum of polar codes, which indicates the distribution of codewords with various weights, is crucial as it provides insight into the performance of codes under Maximum Likelihood (ML) decoding. If the channel is AWGN, then the probability of error can be formulated as:

$$P_e = \sum_{w=d^*(\mathcal{C})}^{N} A_w \mathcal{Q}\left(\sqrt{2wR\frac{E_b}{N_0}}\right) \tag{1.16}$$

where A_w denotes the number of codewords with weight w, $\frac{E_b}{N_0}$ denote the Signal to Noise Ratio (SNR) and $\mathcal{Q}(u)$ is the function defined by $\mathcal{Q}(u) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-\frac{u^2}{2}} \mathrm{du}$. This expression is usually approximated by only keeping the term corresponding to the minimum distance as it is the dominant one as:

$$P_e \approx A^*(\mathcal{C}) \mathcal{Q}\left(\sqrt{2d^*(\mathcal{C})R\frac{E_b}{N_0}}\right) \tag{1.17}$$

where $A^*(\mathcal{C})$ denotes the number of codewords with minimum distance. Many studies were therefore dedicated to the computation of the first element(s) of the distance spectrum of polar codes.
The minimum distance of polar codes is defined in [23] as:

$$d^*(\mathcal{C}) = \min_{i \in \mathcal{I}} 2^{wt(i)} \tag{1.18}$$

where $wt(i)$ denotes the number of ones in the binary expansion of i. This computation is valid for any chosen information set $\mathcal{I}$.
The computation of the number of codewords with minimum weight was introduced in [16] and is only valid for polar codes that are considered to be *Decreasing Monomial Codes (DMC)*. In the particular case of a polar code being a DMC, the number of codewords with minimum weight $A^*(\mathcal{C})$ can be determined as explained in [16]. A novel efficient enumeration of codewords with minimum weight for polar codes with any frozen set is discussed in the next chapters. Figure 1.7 shows the evolution of the minimum distance as well as the number of codewords with minimum weight for a GA and a RM-GA rate-profiling strategies. One can observe that RM-GA rate-profiling strategy exhibit better distance properties than GA strategies. This means that the polar codes under ML decoding with RM-GA rate profiling outperform the polar codes under GA construction.

1.3 On the pre-transformation of polar codes

One limitation of polar codes arises from the fact that, for short and moderate code lengths, polar codes have poor distance properties. The enhancement of the distance properties of polar codes saw a notable stride with the introduction of pre-transformation. Among the pre-transformation techniques, the concatenation of cyclic redundancy check (CRC) with polar codes [24] enables to improve the distance properties and the decoding performance of polar codes. A more recent alternative is to apply precoding before the polar transformation. Thus, *Dynamic Frozen Bits (DFB)* polar codes utilize an upper

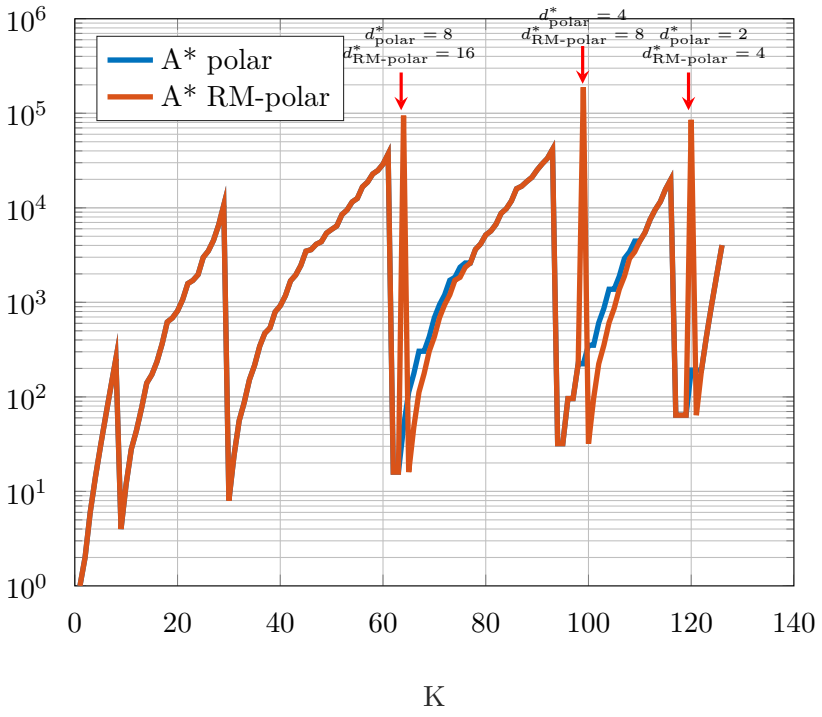


Figure 1.7: Minimum distance and associated number of occurrences for a $(128, 64)$ polar code under polar and RM-polar constructions

triangular transformation on frozen bits [11]. *Polarization-Adjusted Convolutional* (PAC) codes combine a Polar code and a rate-1 convolutional encoder [10]. In this section, the different types of precoding are introduced. Moreover, their impact on the distance properties of precoded polar codes is discussed.

1.3.1 Polarization-Adjusted Convolutional (PAC) codes

PAC codes [10] denoted by PAC$(N, K, \mathbf{g})$ consist in polar codes where the input vector $\mathbf{u}_0^{N-1}$ is obtained by a convolutional transformation using the generator polynomial $\mathbf{g}$ of degree $m-1$ with coefficients $[g_0, g_1, ..., g_{m-1}]$. The encoding scheme for PAC codes is represented in Figure 1.8. It operates as follows:

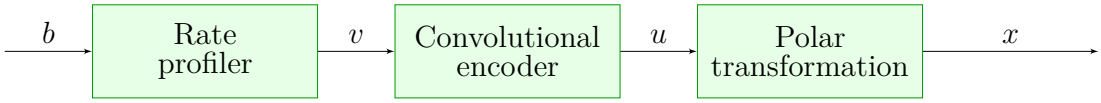


Figure 1.8: PAC code encoding scheme

- The information bit vector $\mathbf{b}$ is first mapped to the vector $\mathbf{v}$ using the chosen rate-profiling

- The vector $\mathbf{v}$ is then transformed thanks to the rate-1 convolutional encoder with coefficients $\mathbf{g}$ in order to obtain $\mathbf{u}$. For $i \in [\![0; N-1]\!]$, i.e. given a vector $\mathbf{v}_0^i$, the associated u_i is obtained as follows:

$$u_i = \mathbf{g}(\mathbf{v}_0^i) = \sum_{j=0}^{m-1} g_j v_{i-j} \tag{1.19}$$

  In other terms, the input element u_i of the polar encoder depend on the $i-1$ previous elements as well as the current element v_i. Conventionally, we consider $g_0 = g_{m-1} = 1$.

- the vector $\mathbf{u}$ is then multiplied by the matrix $\mathbf{G}_N$ in order to obtain the codeword $\mathbf{x}$

The impact of the pre-transformation of PAC codes resides in the fact that the frozen bits are not necessarily equal to 0 anymore, i.e., there can exist $i \in \mathcal{F}$ such that $u_i \neq 0$. From a matrix point of view, the convolution can be seen as a multiplication with an upper-triangular Toeplitz matrix $\mathbf{T}$ given as:

$$\mathbf{T} = \begin{bmatrix} g_0 & g_1 & g_2 & \cdots & g_{m-1} & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & \cdots & \cdots & g_{m-1} & \ddots & \vdots \\ 0 & 0 & g_0 & \cdots & \cdots & \cdots & g_{m-1} & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & 0 & \vdots \\ \vdots & & & \ddots & 0 & g_0 & g_1 & g_2 \\ \vdots & & & & \ddots & 0 & g_0 & g_1 \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & 0 & g_0 \end{bmatrix} \tag{1.20}$$

The overall encoding process can be described as:

$$\mathbf{x} = \underbrace{\mathbf{v}\mathbf{T}}_{\mathbf{u}}\mathbf{G} \tag{1.21}$$

1.3.2 Polar codes with Dynamic Frozen Bits

The main idea behind polar codes with Dynamic Frozen Bits (DFB) [11] is to set the value of frozen bit, initially treated as 0 to an value that depend on the previously determined bits. In other terms, given $i \in \mathcal{F}$, u_i is expressed as:

$$u_i = f_i(u_0, u_1, \cdots, u_{i-1}) \tag{1.22}$$

It means that every frozen bit is expressed as a linear combination of the previously decoded bits.

There are many similarities between PAC codes and polar codes with DFB. Note that while a pre-transformation is applied for every u_i, in the case of polar codes with DFB, only frozen bits are affected with the pre-transformation. Therefore, polar codes with DFB can be seen as PAC codes with a variable generator polynomial for every u_i as follows:

$$\mathbf{g} = \begin{cases} [1], \forall i \notin \mathcal{F} \\ [g_0, ..., g_{i-1}], \forall i \in \mathcal{F} \end{cases} \tag{1.23}$$

1.3.3 Impact of precoding on the distance properties of polar codes

In this section, we give a brief explanation of the impact of pre-transformation on polar codes distance properties to justify the growing interest regarding those techniques. It will be further explored in Chapters 2 and 3. Authors of [25] proved that any pre-transformation that is applied on polar codes using an upper triangular matrix does not lower the code's overall minimum distance. In other words, given a polar code $\mathcal{C}(N, K, \mathcal{F})$ and its pre-transformed version $\mathcal{C}_{PT}(N, K, \mathcal{F}, \mathbf{T})$ then the following equation is verified:

$$d^*(\mathcal{C}) \leq d^*(\mathcal{C}_{PT}) \tag{1.24}$$

In other words, a pre-transformation cannot lower the minimum distance of a polar code, yet, it can increase it.
It is also noted in [25] that for a polar code and a pre-transformed polar code with identical minimum distances, the number of codewords with minimum weight of a pre-transformed polar code is lower compared to the polar code.

1.4 Decoding of pure and pre-transformed polar codes

1.4.1 Successive Cancellation List decoding

1.4.1.1 Case of pure polar codes

When reviewing the SC decoding outlined in Section 1.1.4, it appears that its primary drawback lies in the hard decisions made during the decoding of each $\hat{u}_i$. This signifies that in the event of an error occurring during the decoding process of a bit, no error recovery can be undertaken. The Successive Cancellation List (SCL) decoder introduced in [8] has of main objective to delay the hard decision taken for each $\hat{u}_i$. Instead of solely making the decision based on the LLR value as explained in equation (1.9), the two decoding possibilities are considered and kept in a list of size L. Hence each path kept by the SCL algorithm is updated according to the SC decoding process. This can be explained by the fact that at each decoding stage i, and for $l \in [1, L]$, the probability $P(\hat{\mathbf{u}}_0^i[l]|\mathbf{y})$ is evaluated such as:

$$P(\hat{\mathbf{u}}_0^i[l]|\mathbf{y}) = \prod_{j=0}^{i} P(\hat{u}_j[l]|\hat{\mathbf{u}}_0^{j-1}[l], \mathbf{y}) \tag{1.25}$$

Note that in the case of ML decoding, the probability described in Equation (1.25) is maximised for each $\hat{\mathbf{u}}_0^i$. This means that the ML decoder can be viewed as an SCL decoder with L being large enough to enable the exploration of all the 2^K paths at each decoding stage. Actually, the SCL decoder only explores a reduced number L of paths instead of exploring all the paths to mitigate impractical computational complexity.

Therefore, each decoding path splits into two children paths (in the case where the considered bit is an information bit). A metric is introduced to penalize one decoding path at the expense of the other one. The algorithm then selects the best L candidates having the lowest path metrics. The metric expression is updated at each decoding stage and is based on the logarithm of Equation (1.25) to enable a recursive update. In particular, the metric of the l^{th} path at the decoding stage i $m_i(\hat{\mathbf{u}}_0^i[l])$ is defined as:

$$m_i(\hat{\mathbf{u}}_0^i[l]) = -\log(P(\hat{\mathbf{u}}_0^i[l]|\mathbf{y})) \tag{1.26}$$

By using Equation (1.25), $m_i(\hat{\mathbf{u}}_0^i[l])$ can also be expressed as:

$$\begin{aligned} m_i(\hat{\mathbf{u}}_0^i[l]) &= -\sum_{j=0}^{i} P(\hat{u}_j[l]|\hat{\mathbf{u}}_0^{j-1}[l], \mathbf{y}) \\ &= m_{i-1}(\hat{\mathbf{u}}_0^{i-1}[l]) + \mu_i(l) \end{aligned} \tag{1.27}$$

where $\mu_i[l] = -P(\hat{u}_i[l]|\hat{\mathbf{u}}_0^{i-1}[l])$ denotes the branch metric. It has been proven in [26] that $\mu_i[l]$ can also be expressed as:

$$\mu_i[l] = \log(1 + e^{-(1-2\hat{u}_i[l])L_i}) \tag{1.28}$$

In the case where $\text{sign}((1 - 2\hat{u}_i[l])) = \text{sign}(L_i)$, $\mu_i(l) = \log(1 + e^{-|L_i|}) \approx 0$. Similarly, in the other case, $\mu_i(l) \approx e^{|L_i|}$. This leads to a simplification in the metrics update process that can be formulated as follows:

$$m_i(\hat{\mathbf{u}}_0^i[l]) = \begin{cases} m_{i-1}(\hat{\mathbf{u}}_0^{i-1}[l]) & \text{if } \text{sign}((1 - 2\hat{u}_i(l))) = \text{sign}(L_i) \\ m_i(\hat{\mathbf{u}}_0^i[l]) + |L_i| & \text{otherwise} \end{cases} \tag{1.29}$$

In other words, the paths where the decision on $\hat{u}_i$ is different from the LLR decision are penalized with $|L_i|$. As $|L_i|$ is an indication on the bit's reliability, paths that are not compliant to the LLR decision are heavily penalized when the bit is highly reliable and vice versa. When all the bits are processed, the SCL algorithm selects the path with the smallest metric. Figure 1.9 represents the SCL decoding process for a polar code with a list size $L = 4$ (Note that only the information bits are displayed). We can see that, at each decoding stage, only the 4 candidates with the least metric are kept in the list and that at the end of decoding, the path with the least metric is chosen.

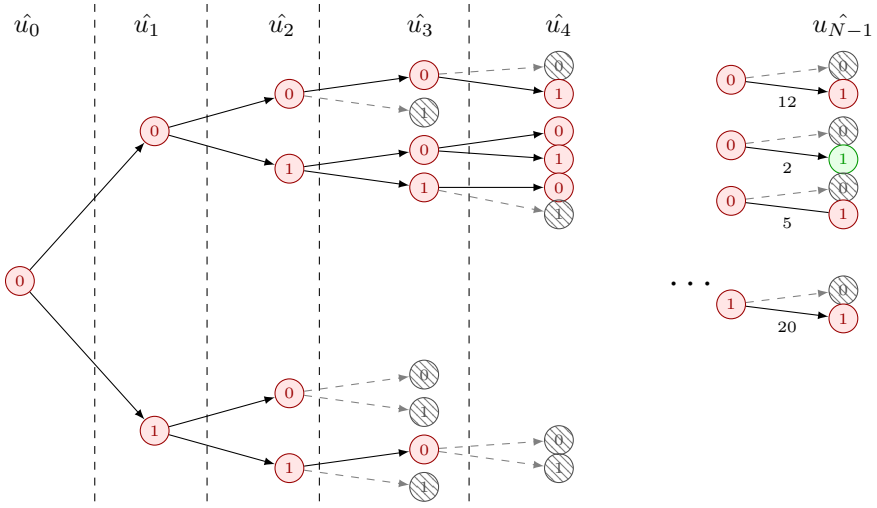


Figure 1.9: Successive Cancellation List decoding with $L = 4$

1.4.1.2 SCL decoding to pre-transformed polar codes

In this section, we introduce the mechanism of SCL decoding for pre-transformed polar codes. The results are presented for PAC codes, but can also be generalised to polar codes

with DFB following the previously mentioned relation between PAC codes and polar codes with DFB in Equation (1.23). Algorithm 3 describes the overall SCL decoding process for PAC codes that was introduced in [27]. One can note that this algorithm is also valid for polar codes. In fact, polar codes are PAC codes with $\mathbf{g} = [1]$. It operates identically as the SCL decoding for pure polar codes. The only difference is that instead of only storing the different vectors $\hat{\mathbf{u}}$, the paths $\hat{\mathbf{v}}$ has also to be stored in memory.

1.4.2 Concatenation with a Cyclic Redundancy Check

It has been observed in [8] that in most of the cases when the SCL decoder fails into recovering the initially transmitted message, the transmitted path is within the final list with very high probability. This led [8] and [24] to propose to concatenate a Cyclic Redundancy Check (CRC) with a polar code. In addition to increasing the minimum distance of the code, the CRC helps the SCL decoder to find the transmitted message. Cyclic Redundancy Code (CRC) Aided SCL decoder, denoted by CA-SCL, comes therefore as an improvement of SCL that consists of introducing a CRC within the information vector $\mathbf{u}$ in order to check if a path is "valid" or not. CA-SCL algorithm chooses the candidate with a valid CRC first and if no or several candidates have a valid CRC, the one with the lowest SCL metric is chosen. The encoding process for CA-SCL decoding is represented in Figure 1.10

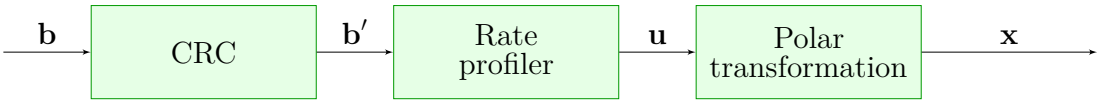


Figure 1.10: Encoding scheme for polar codes with CRC

1.4.3 Sequential decoding of polar codes

Sequential decoding, initially proposed in [28], is a depth-first search heuristic specifically designed for decoding arbitrary tree codes. It differs from the list decoder with its ability to go backward during the path search process. In fact, in the case of an SCL decoder, if a path $\mathbf{u}_0^i \in \mathbb{F}_2^{i+1}$ is discarded at a decoding stage i, it cannot be in the list again unlike for the sequential decoding. This further suggests that, unlike the SCL decoder, the computational complexity of a sequential decoder is variable and contingent upon both the properties of the code and the parameters of the channel. The Stack algorithm [29] and the Fano algorithm [30] stand out as two of the most renowned sequential decoder algorithms. Sequential decoding was initially introduced for polar codes under the name Successive Cancellation Stack (SCS) in [31, 32]. Another variant, which varies in metric

Algorithm 3: SCLDecode($N, \boldsymbol{\lambda}, \mathcal{F}, L, \mathbf{g}$)

Input: Code length N, Channel LLRs $\boldsymbol{\lambda}$, frozen set $\mathcal{F}$, list size L, generator polynomial $\mathbf{g}$

Output: Bit vector estimation $\hat{\mathbf{u}}$

$\mathcal{L} \leftarrow \{0\}$ `/* List to store the different paths */`
for $i \leftarrow 0$ **to** $N-1$ **do**
  if $i \in \mathcal{F}$ **then**
    for $l \leftarrow 1$ **to** $|\mathcal{L}|$ **do**
      $L_i[l] \leftarrow \text{ComputeLLR}(N, \boldsymbol{\lambda}, \hat{\mathbf{u}}_0^{i-1})$
      $\hat{v}_i[l] \leftarrow 0$
      $\hat{u}_i[l] = \mathbf{g}(\hat{\mathbf{v}}_0^i[l])$
      $m_i(\hat{\mathbf{u}}_0^i[l]) \leftarrow \text{ComputeMetric}(m_{i-1}(\hat{\mathbf{u}}_0^{i-1}[l]), L_i[l], \hat{u}_i[l])$
    end
  end
  else
    $\mathcal{L} \leftarrow \mathcal{L} \bigcup \mathcal{L}'$
    `/*` $\mathcal{L}'$ `is a copy of` $\mathcal{L}$ `*/`
    for $l \leftarrow 1$ **to** $|\mathcal{L}|$ **do**
      $L_i[l] \leftarrow \text{ComputeLLR}(N, L_0^{N-1}, \hat{\mathbf{u}}_0^{i-1})$
      $[v_i[l], v_i[l']] \leftarrow [0, 1]$
      $\hat{u}_i[l] = \mathbf{g}(\hat{\mathbf{v}}_0^i[l])$
      $\hat{u}_i[l'] = \mathbf{g}(\hat{\mathbf{v}}_0^i[l'])$
      $m_i(\hat{\mathbf{u}}_0^i[l]) \leftarrow \text{ComputeMetric}(m_{i-1}(\hat{\mathbf{u}}_0^{i-1}[l]), L_i[l], \hat{u}_i[l])$
      $m_i(\hat{\mathbf{u}}_0^{i-1}[l']) \leftarrow \text{ComputeMetric}(m_{i-1}(\hat{\mathbf{u}}_0^{i-1}[l']), L_i[l'], \hat{u}_i[l'])$
      if $|L| > L$ **then**
        $L \leftarrow$ Discard L paths with the highest path metrics
      end
    end
  end
  $\hat{\mathbf{u}}_0^{N-1} \leftarrow$ Remaining path with the lowest metric
  return $\hat{\mathbf{u}}_0^{N-1}$
end
subroutine `ComputeMetric`(m, L, $\hat{u}$)
  if $\hat{u} = \frac{1}{2}(1 - sign(L))$ **then**
    return m
  end
  else
    return $m + |L|$
  end

definition, was subsequently explored in [33, 34]. It's worth noting that [33] and [34] also investigated the implications of sequential decoding for polar codes with DFB.
When introducing PAC codes in [10], Arikan showed that using a Fano decoder for PAC codes, the FER can reach the finite-length capacity bound [35]. Several studies, including [36] and [37], have delved deeper into the properties of Fano decoding for PAC codes. This algorithm can however have a very high computational complexity in the worst case.

The Fano algorithm stands out as one of the most practical sequential decoding algorithms as it is able to analyze only a single path at a time. It eliminates the need to store more than one path and its associated metric. One fundamental parameter is its threshold defined as that can only take discrete values spaced apart by increments of Δ, i.e. $0, \pm\Delta, \pm 2\Delta$.... The explored path metric is denoted by Γ.

Essentially, the Fano algorithm continues to explore further along (Forward search) a specific path as long as the metric value of the path continues to increase. In other words, in order to continue a forward search, at each decoding stage, and in the case of an information bit, the path with the higher metric Γ is kept. The chosen path not only needs to have the highest metric but also its metric must surpass the fixed threshold. During forward searching, when the path metric is increasing, the threshold is tightened (increased by Δ) and the condition $\Gamma > T$ has to be satisfied.

If the condition $\Gamma > T$ is no longer satisfied, then the algorithm revisits earlier nodes along previously traversed paths and explores alternative pathways originating from them in order to potentially find another path with a better metric. This is called Backward search. In contrast to forward searching, during backward searching, the threshold is decreased by Δ. This process is conceived in a way that no node is searched forward more than once with the same threshold, as it must always be smaller than the previously used value.

1.5 Performance of pure and pre-transformed polar codes under different decoding schemes and constructions

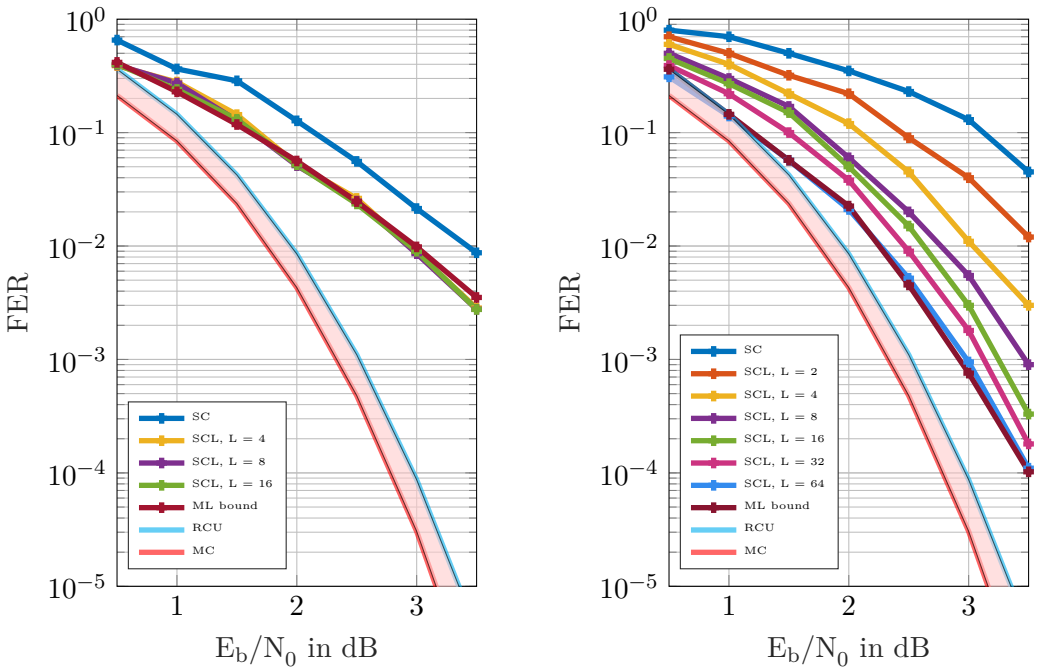
In this section, the performance outcomes of polar codes across various parameters such as block length, construction method, pre-transformation, decoding algorithm, list size, etc., are presented. The aim here is to conduct a comparative analysis of polar code performances under different rate-profiling schemes, assess the list size required for each configuration to approach the ML bound, illustrate the influence of precoding on polar

code performances, and examine the proximity of actual configurations documented in the literature to the theoretical bounds across different codelengths.

1.5.1 Short-length codes

1.5.1.1 Pure polar codes

In this section, we will focus on the $(128, 64)$ code as an example for short codes. Figures 1.11b and 1.11a show the evolution of the Frame Error Rate (FER) for a $(128, 64)$ polar codes under RM and 5G [9] constructions respectively and under SC decoding and SCL decoding with different list sizes. We also represent the Truncation Union Bound (TUB) detailed in Equation (1.17) as an approximation to the performance of ML decoding. We also represent the Polyanskyi Poor-Verdu (PPV) [35] Meta Converse (MC) bound and the Random Coding Union (RCU) bound. The RCU and MC bounds define respectively an upper and a lower bound on the optimal achievable performance for a finite block-length code. The presented RCU and MC bounds are based on the saddle points approximation detailed in [38] and [39]. The implementation provided in [40] was used.



(a) 5G construction (b) RM construction

Figure 1.11: Polar codes performance under SC and SCL decoding

We can see from Figures 1.11b and 1.11a that under ML decoding, polar codes under RM construction widely outperform polar codes under 5G construction. Table 1.1 displays

Construction	A_8	A_{12}	A_{16}
5G	304	768	161528
RM	0	0	94488

TABLE 1.1: Polar codes distance properties under 5G and RM construction

Type	A_8	A_{12}
Pure polar	176	0
CRC polar	0	133

TABLE 1.2: Polar codes distance properties under for pure and CRC polar codes

the first elements of the distance spectrum of polar codes under 5G and RM constructions. Those elements can be obtained using [24] or methods we will introduce in Chapter 2. It can be observed that RM polar codes have a minimum distance that is twice larger than the one for 5G polar codes. This explains the gap between the performance of both codes under ML decoding (denoted by TUB in both figures). On the other hand, it is observed that while polar codes under RM constructions need a larger list size to achieve the ML performance (approximately for $L = 64$), polar codes under 5G construction achieve the ML performance even for $L = 4$ since the decoding performances for $L = 4$ and larger list size superimpose with the ML performances. Under SC decoding and SCL decoding for L up to 2, it is however observed that polar codes under 5G construction outperform polar codes under RM construction.This illustrates a trade-off between a code's performance under ML decoding and the required list size to attain such performance levels. In other words, a code construction solely based on the code's distance properties yields ML decoding performances close to the RCU bound. However, it might necessitate larger list sizes to achieve ML performance using a list decoder.
Figure 1.12 presents the performance of a $(128, 64)$ polar code under DE-GA construction with $\frac{E_b}{N_0} = 4.5$dB. The performances are shown under SCL decoding for a list size $L = \{8, 16, 32\}$ (Similarly to polar codes under 5G construction, the ML bound is rapidly reached when increasing L) and for a $(128, 64)$ polar code concatenated with a CRC of length 7 (the code presented in [41]) for different list sizes.

As illustrated in Figure 1.12, concatenating a CRC with a polar code enhances the code's performance under similar list decoding parameters. This improvement primarily stems from the enhanced distance properties achieved by the polar code with CRC concatenation. Table 1.2 displays the minimum distance for both a pure polar code and a polar code concatenated with a CRC. This table shows that a polar code with CRC exhibits a higher minimum distance compared to a pure polar code. It is also observed that the performance of polar codes with CRC is improved for larger list size. A $(128, 64)$ polar code exhibits better performance under CA-SCL decoding with a list

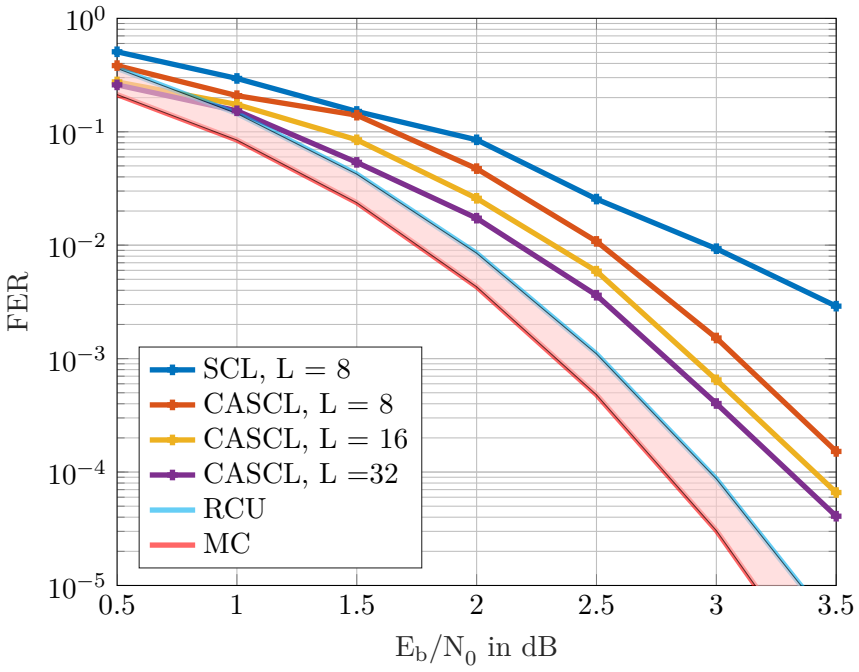


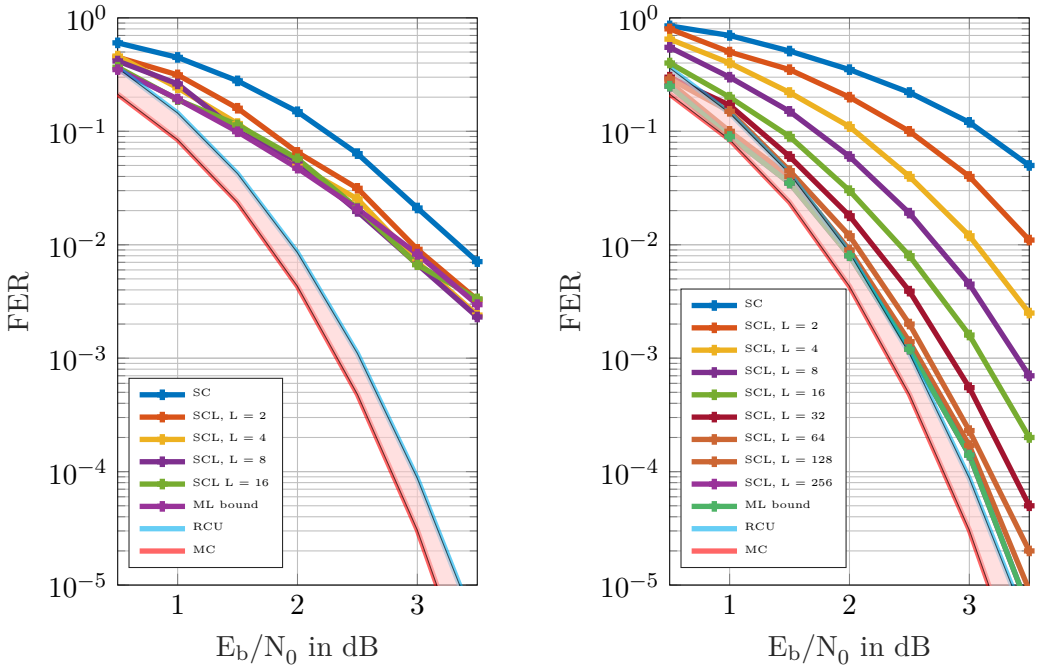
Figure 1.12: Polar codes performance under SCL and CA-SCL decoding and GA construction

$L = 32$ than a polar code under RM construction and ML decoding. The performance improvement achieved by concatenating a CRC to the polar code primarily stems from the enhancement of its distance properties. However, a larger list size is required in order to achieve the ML performances.

1.5.2 PAC codes

In this section, we elaborate on the performance of PAC codes to evaluate the influence of precoding on the performance of polar codes. Figures 1.13a and 1.13b show the evolution of the FER for a $(128, 64)$ PAC code using the polynomial $\mathbf{g} = [1, 0, 1, 1, 0, 1, 1]$ under RM and 5G constructions respectively.

The decoding methods employed include SC decoding, SCL decoding with different list sizes and Fano decoding. Similarly to pure polar codes, the TUB for PAC codes under 5G construction is lower by several orders of magnitude compared to PAC codes under RM construction. Additionally, it's noteworthy that like polar codes under RM construction, PAC codes under RM construction require a larger list size (approximately $L = 256$) to attain the same performance as Maximum Likelihood (ML) decoding in high Signal-to-Noise Ratio (SNR) regimes. In contrast, for polar codes under 5G construction, a list size



(a) 5G construction (b) RM construction

Figure 1.13: PAC codes performance under SC and SCL decoding

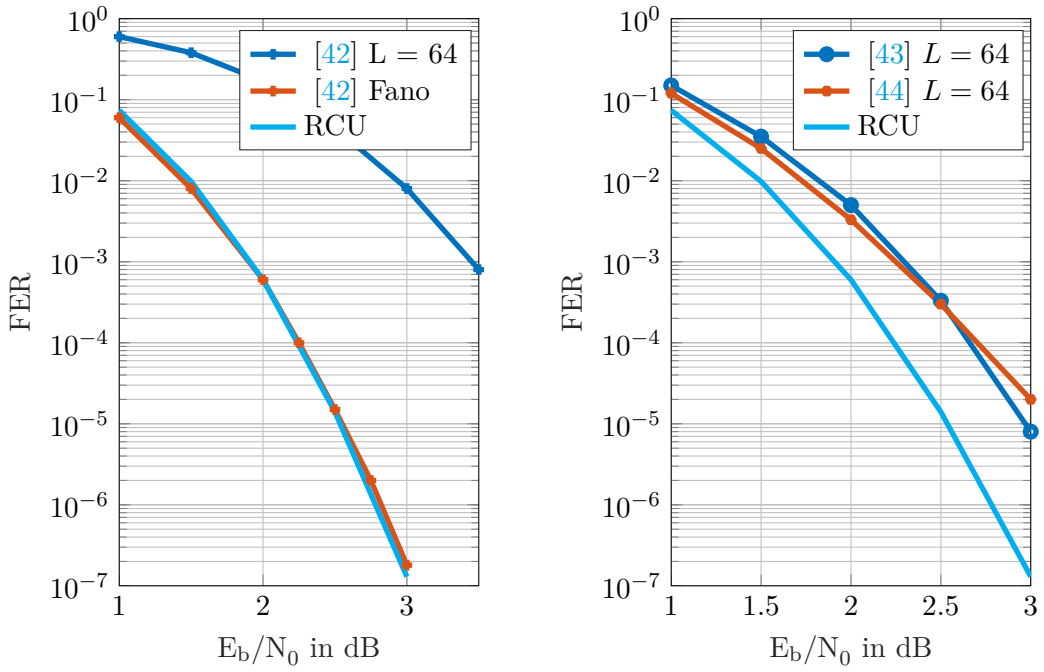
Construction	Type	A_8	A_{16}
5G	Pure	304	161528
	PAC	256	76056
RM	Pure	0	94488
	PAC	0	3120

TABLE 1.3: Pure polar and PAC codes distance properties

of $L = 4$ is sufficient to achieve ML performance. It is important to highlight that the $(128, 64)$ PAC code, when decoded using Fano decoding or list decoding with $L = 256$, achieves performance that closely approaches the RCU bound. Regarding the difference in decoding performance between polar codes and PAC codes, PAC codes exhibit superior performance. This superiority can be explained by the pre-transformation of polar codes, as discussed in section 1.3.3, which reduces the number of codewords with minimum weight. Table 1.3 provides insight into the number of codewords with minimum weight for both polar codes and PAC codes under 5G and RM constructions. From Table 1.3, it can be observed that the reduction in the number of low-weight codewords is significantly higher in the case of RM construction compared to 5G construction. This explains the larger performance gap observed between polar codes and PAC codes under RM construction.

1.5.3 Larger length codes

In the preceding section, we observed that PAC codes under RM construction exhibit performances that approach the RCU bound for a moderate to large list size ($L = 128$). In this section, our attention shifts to the latest advancements in polar codes and PAC codes for longer codes to verify if the previous results hold true as well. Figure 1.14a illustrates the performance of a $(256, 128)$ PAC code using the constructions outlined in [42] that used a genetic algorithm to enhance the distance properties of PAC codes, employing Fano decoding and SCL decoding. The figure indicates that under Fano decoding, the



(a) [42] under SCL and Fano decoding (b) [43] and [44] under SCL decoding

Figure 1.14: $(256, 128)$ PAC codes performance

performance of the PAC code closely matches the RCU bound. However, when applying a SCL decoding with a moderate list size of $L = 64$, the performance notably degrades. The authors of [43] introduced PAC codes that exhibit superior performance compared to those in [42] when decoded using SCL decoding with a moderate list size. However, there remains a discernible gap to the RCU bound in terms of performance. The performance of those codes under SCL decoding with $L = 64$ are represented in Figure 1.14b.

The same issue persists with codes of higher code block lengths. Figure 1.15 illustrates some of the most performant pre-transformed polar codes under SCL decoding with various list sizes.

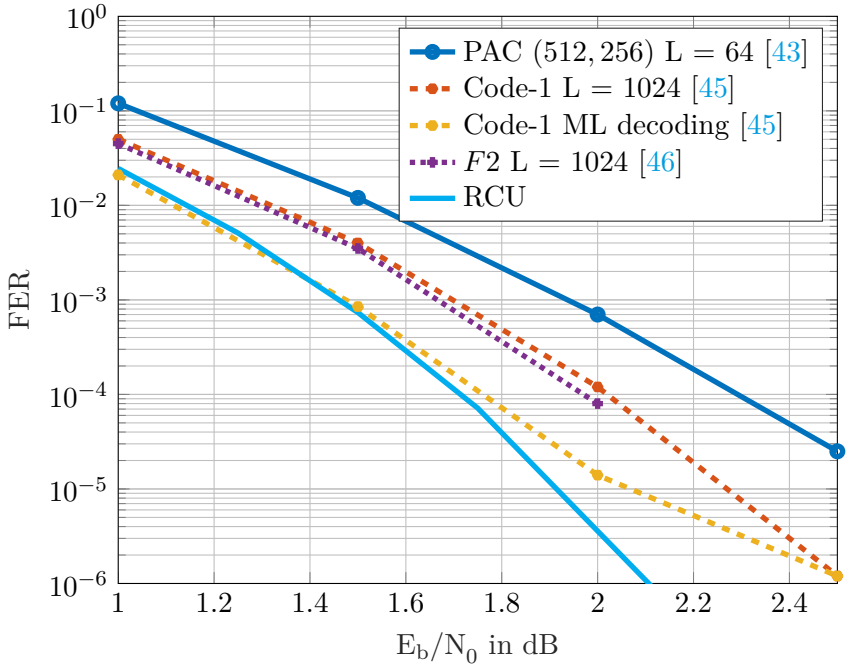


Figure 1.15: $(256, 128)$ pre-transformed polar codes performance

One can see that, even with large list sizes, there remains a discernible gap to the RCU bound. This phenomenon applies to longer codes as well. We will delve into this issue in greater detail in Chapter 4.

1.6 Problematics and conclusions

In this section, we review the various elements that were introduced in this chapter and delve into the issues that have been addressed in the upcoming chapters.

In this chapter, we delved into the fundamental principles of polar codes, which stand out as the first class of deterministic error-correcting codes proven to achieve channel capacity for a B-DMC under SC decoding. Furthermore, we provided insights into the diverse decoding algorithms available for polar codes, offering a balance between decoding performance and computational complexity. The diverse decoding techniques enhanced the performance of polar codes for moderate codelengths, rendering them competitive with LDPC codes and turbo-codes. This positions polar codes as a compelling area for exploration. Additionally, we examined how polar codes can be further optimized through various pre-transformation techniques and alternative construction methods, thereby influencing their performance, particularly in terms of their distance properties.

As of the present moment, the literature covers short length precoded polar codes with specific code rates, which approach the RCU bound under SCL decoding with moderate list sizes. The first limitation is that the code construction is only optimal for certain code rates. Moreover, the gap to the RCU bound for polar codes with higher codelengths remains non-negligible under SCL decoding with moderate list sizes, prompting further research efforts in this direction. The aim is to develop code constructions with ML performances that closely approach the RCU bound, while simultaneously avoiding the need for excessively large list sizes to achieve such performances, as this would entail significant computational complexity.

Since the distance properties of polar codes are what drive the performances of polar codes under ML decoding, the aim is to understand the interaction between the list decoder and the distance properties of the code. This is because the polar constructions based on the channels reliability are optimal under SC decoding. However, it's not guaranteed that they represent the optimal choice under decoding algorithms that explore more than one path such as the list decoder. This explains why under such constructions, the ML bound is reached even with low list sizes, but the gap between their ML bound and the RCU bound is large. On the other hand, polar codes with optimal distance properties perform well under decoding algorithms that approximate ML decoding but have very high computational complexity. However, under SCL decoding with moderate list sizes, they may exhibit poor performances.

In the next chapter, we introduce a novel low-complexity method for computing the minimum distance properties of both pure and precoded polar codes. This method is

then extended to punctured and shortened pure and precoded polar codes in Chapter 3. The final chapter is dedicated to presenting a method for efficiently constructing rate profiles for short length codes, encompassing all possible code rates. Additionally, it addresses the challenge of determining the required list size to achieve ML decoding under a specific code construction.

2 On the distance properties of pure and pre-transformed polar codes

This chapter focuses on the determination of the minimum distance properties of pure, pre-transformed and shortened polar codes. Our contributions consist in two parts: 1) Computation of the distance properties of *polar cosets*, 2) Generalisation to the minimum distance properties or the partial weight spectrum of pure and pre-transformed polar codes.

2.1 Context

Polar codes as designed by Arikan [1] have poor performance at finite length under SC decoding. As an approximation of ML decoding, the SCL decoder enables to partially close this gap if the list size is large enough. However, the mediocre distance properties of polar codes prevent them from better performance for medium / low lengths. In order to improve the distance properties, the coding structure has to be modified.

A first option, used in the 5G standard, is to add a CRC as an outer code [9]. This increases the distance of the code and allows the SCL decoder to discard candidate codewords at the end of the decoding process. A more recent alternative is to apply a rate-1 precoding before the polar transformation. As such, the *polar codes with dynamic frozen bits* is based on an upper triangular transformation on the frozen bits [11]. In the case of *Polarization-Adjusted Convolutional* (PAC) codes, a rate-1 convolutional encoder [47] is considered.
Considering all these types of polar codes, one challenge is to be able to calculate their distance properties regardless of the code length and the structure of the frozen bits. The knowledge of the distance properties of polar codes can either help building new rate-profiling strategies for polar codes and or rapidly estimate the decoding performance of the ML decoder at high SNR.
The computation of the minimum distance of polar codes was first proposed in [23]. An explicit formula for the minimum distance is presented and can be used for pure polar codes regardless of the frozen bit set strategy. In [16], the number of minimum weight codewords is computed under the assumption that the polar code is a decreasing monomial code. A more general approach is described in [24] where Monte Carlo simulation of an SCL decoder with very large list size allows to estimate the partial distance of polar codes. However, this approach is very computational complex as the code length grows. Besides, this method being not deterministic, there is no guarantee to find all the searched codewords. In the case of PAC codes, it is also possible to use the complex Monte Carlo approach [48].
A method to compute the polar spectrum of polar codes was introduced in [49]. Then, in [50], a deterministic algorithm to compute the weight distribution for polar codes has been introduced. However, its high computational complexity made it impractical even for moderate code lengths. Some computational complexity reduction can only be achieved for specific frozen bit sets. It however cannot be considered for codes with a length greater than 128.
[51] provided a complexity-reduced algorithm to enumerate the codewords with minimum weights for polar codes and a method to compute the codewords with minimum weights for polar codes with CRC. The complexity reduced algorithm is however only restricted

to specific constructions of frozen bit sets and can only be applied to polar codes, making the computational complexity of the minimum distance of polar codes with CRC very high.

In [14], the low-weight codewords of polar and pre-transformed polar codes are enumerated based on a recursive decomposition. Although very general, this approach becomes complex for pre-transformed polar. Indeed, for pre-transformed polar codes, the decomposition is less favorable than for regular polar codes.

Other probabilistic techniques, such as those outlined in [52] and [53], aid in accurately estimating the complete weight distribution of polar codes. However, these approaches do not apply to pre-transformed polar codes. Reference [54] introduced a method to compute the average partial weight spectrum of pre-transformed polar codes and is therefore not deterministic. Additionally, works by [55] and [56] focused on calculating the partial weight spectrum of polar codes. Reference [57] proposed a method to compute the entire spectrum for polar codes with low or high rates. These methodologies are specific to polar constructions that are decreasing monomial and cannot be extended to pre-transformed polar codes. [58] followed by [59] presented a low-complexity technique for computing the minimum distance properties of pre-transformed polar codes, but limited to scenarios where the pre-transformed polar codes conserve the same minimum distance as pure polar codes. Table 2.1 sums up the different advantages and limitations of the previously discussed references. In this chapter, a low-complexity algorithm in

	Pre-transformation	Partial or whole weight spectrum	Deterministic	Complexity	Rate-profiling
[50]	✓	✓	✓	Very high	All
[58]	✓	✗	✓	Moderate to low	Only DMC
[54]	Only averaged on all possible pre-transformations	✓	✗	Moderate to low	All
[53], [52]	✗	✓	✗	Moderate to low	All
[55], [56], [57]	✗	✓	✓	Moderate to low	Only DMC
[51]	✓	✗	✓	High	All
[24], [48]	✓	✓	✗	Very high	All
[14]	✓	✓	✓	High	All
[59]	✓	✗	✓	Moderate to low	Only for PT that keep the same d^*

TABLE 2.1: Distance properties determination methods: advantages and limitations

terms of the number of calculations capable of computing the minimum distance and its associated number of occurrences for pure and pre-transformed polar codes is proposed.

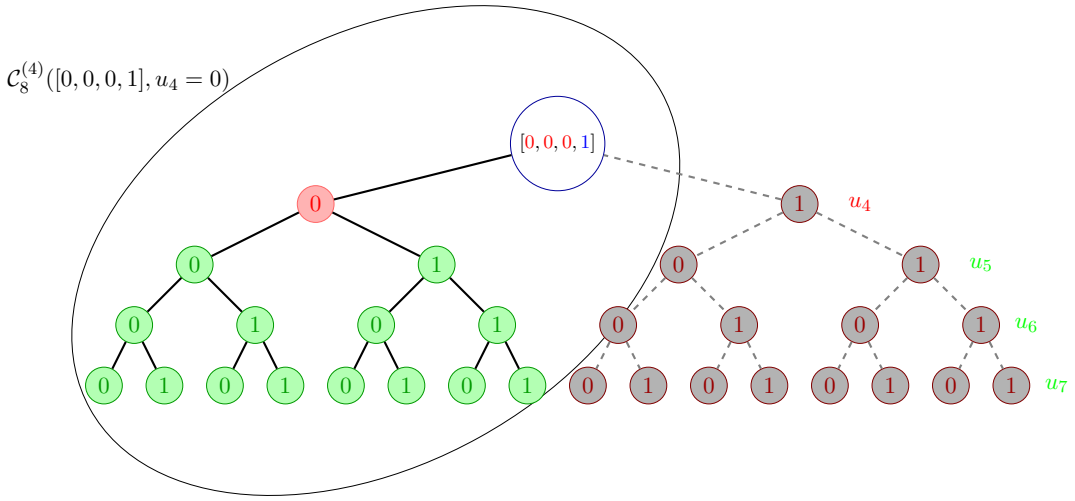


Figure 2.1: Polar coset code $\mathcal{C}_N^{(3)}(0,0,0,1)$ with $N = 8$ and $\mathcal{I} = \{3,5,6,7\}$

2.2 Graph computation of the minimum distance and associated number of occurrences of polar cosets

2.2.1 Polar cosets

As in [50], given $\mathbf{u}_0^{i-1} \in \mathbb{F}_2^{i-1}$ and $u_i \in \mathbb{F}_2$, a polar coset $\mathcal{C}_N$ can be defined as:

$$\mathcal{C}_N(\mathbf{u}_0^i) = \{[\mathbf{u}_0^i, \mathbf{u}_{i+1}^{N-1}]\mathbf{G} | \mathbf{u}_{i+1}^{N-1} \in \mathbb{F}_2^{N-i-1}\} \tag{2.1}$$

A polar coset $\mathcal{C}_N(\mathbf{u}_0^i)$ thus describes the codewords' space generated by the prefix $\mathbf{u}_0^i$ without taking into consideration the frozen bits contained in $\mathbf{u}_{i+1}^{N-1}$.

Example 2.2.1. *Figure 2.1 represents the different* $\mathbf{u}_0^7$ *with* $\mathbf{u}_0^3 = [0,0,0,1]$ *and* $\mathbf{u}_4^7 \in \mathbb{F}_2^4$ *used to generate the polar coset* $\mathcal{C}_N([0,0,0,1])$ *for a* $(8,4)$ *polar code. It can be seen from this figure that we consider both cases where* $u_4 = 0$ *and* $u_4 = 1$ *even though* u_4 *is considered to be frozen.*
We can also see from the figure that the coset $\mathcal{C}_8([0,0,0,1,0])$ *is included in the coset* $\mathcal{C}_N([0,0,0,1])$ *or more generally given* $[u_0, ..., u_{i-1}], u_i$ *and* u_{i+1}, $\mathcal{C}_N(\mathbf{u}_0^{i+1}) \subset \mathcal{C}_N(\mathbf{u}_0^i)$.

This can be generalised to:

$$\mathcal{C}_N(\mathbf{u}_0^{N-1}) \subset \mathcal{C}_N(\mathbf{u}_0^{N-2}) \subset \mathcal{C}_N(\mathbf{u}_0^1) \subset ... \subset \mathcal{C}_N(u_0) \tag{2.2}$$

We will demonstrate in the next section that for any polar coset $\mathcal{C}_N(\mathbf{u}_0^i)$, we are able to compute its minimum weight and related number of occurrences.

2.2.2 Computation of the minimum distance properties of a polar coset

In the following, we will denote by $w^*(\mathcal{C}_N(\mathbf{u}_0^i))$ the minimum weight of a polar coset $\mathcal{C}_N(\mathbf{u}_0^i)$ and by $A^*(\mathcal{C}_N(\mathbf{u}_0^i))$ its associated number of occurrences, i.e. the number of codewords of a coset having a weight $w^*(\mathcal{C}_N(\mathbf{u}_0^i))$.
Let $\mathcal{C}_N(\mathbf{u}_0^i)$ define a polar coset. The objective is to calculate, with linear complexity, both the weight $w^*(\mathcal{C}_N(\mathbf{u}_0^i))$ and its associated number of occurrences $A^*(\mathcal{C}_N(\mathbf{u}_0^i))$.
We will focus in a first time on the computation of the minimum weight of polar cosets. In a second time, we show that it is possible to compute the minimum weight and its associated number of cosets simultaneously.

2.2.2.1 Computation of the minimum weight of polar cosets

The aim of this section is to compute the minimum weight of polar cosets thanks to the structure of the polar factor graphs. This will lead to an efficient computation with the rules introduced in [60].

The first step consists into expressing the minimum weight of a coset in a way that enables its factorization as described in [60].

By definition, we have:

$$\begin{aligned} w^*(\mathcal{C}_N(\mathbf{u}_0^{i-1}, u_i)) &\triangleq \min_{\mathbf{u}_{i+1}^{N-1} \in \mathbb{F}_2^{N-i-1}} w([\mathbf{u}_0^{i-1}, u_i, \mathbf{u}_{i+1}^{N-1}]G) \\ &= \min_{\mathbf{u}_{i+1}^{N-1} \in \mathbb{F}_2^{N-i-1}} w(\mathbf{u}_0^{i-1}G_0^{i-1} \oplus u_i G_i \oplus \mathbf{u}_{i+1}^{N-1}G_{i+1}^{N-1}) \end{aligned} \tag{2.3}$$

where G_0^{i-1}, G_i and G_{i+1}^{N-1} respectively denote the upper i first rows, the i^{th} row and the $(N-i)$ last rows of $\mathbf{G}$, respectively.

Given $\mathbf{p} = \mathbf{u}_0^{i-1}G_0^{i-1}$, $\mathbf{m} = u_i G_i$ and $\mathbf{s} = \mathbf{u}_{i+1}^{N-1}G_{i+1}^{N-1}$, (2.3) can be reformulated as:

$$w^*(\mathcal{C}_N(\mathbf{u}_0^{i-1}, u_i)) = \min_{\substack{\mathbf{s} = \mathbf{u}_{i+1}^{N-1}G_{i+1}^{N-1} \\ \mathbf{u}_{i+1}^{N-1} \in \mathbb{F}_2^{N-i-1}}} w(\mathbf{p} \oplus \mathbf{m} \oplus \mathbf{s}) \tag{2.4}$$

Let $\mathbf{x} = \mathbf{p} \oplus \mathbf{m} \oplus \mathbf{s}$. In this section and the following, we will assume the following hypothesis:

Hypothesis 2.2.2. *In the case of polar codes, there exists an extented parity check matrix $H^{(i)^T}$ associated to G_{i+1}^{N-1} and $\mathbf{v} \in \mathbb{F}_2^d$ that verifies*

$$[\mathbf{x}, \mathbf{v}, u_i]H^{(i)T} = \boldsymbol{0} \quad if \quad \exists \quad \mathbf{u}_{i+1}^{N-1} \in \mathbb{F}_2^{n-i-1} | \mathbf{s} = \mathbf{u}_{i+1}^{N-1} G_{i+1}^{N-1} \tag{2.5}$$

and such that the decoding graph of u_i is tree like. $\mathbf{v}$ denote the hidden variable nodes that result from the extension of the parity matrix.

This hypothesis is verified in the case of pure polar codes as it was showed in [61] but can also be generalised to different kernels as long as their associated decoding factor kernels are tree-like.
Equation 2.4 can therefore be expressed as:

$$w^*(\mathbf{p}, u_i) = \min_{\substack{\mathbf{s} \in \mathbb{F}_2^N \\ \mathbf{v} \in \mathbb{F}_2^d}} \sum_{j=0}^{N-1} w(p_j \oplus m_j \oplus s_j) - \log(\mathbb{1}([\mathbf{x}, \mathbf{v}, u_i]H^{(i)T} = \mathbf{0})) \tag{2.6}$$

The term $-\log(\mathbb{1}([\mathbf{x}, \mathbf{v}, u_i]H^{(i)T} = \mathbf{0}))$ ensures that we only consider codewords that belong to the code. Specifically, $-\log(\mathbb{1}([\mathbf{x}, \mathbf{v}, u_i]H^{(i)T} = \mathbf{0})) = +\infty$ if $\mathbf{x}$ is not a codeword of the code.

Since $\mathbb{1}([\mathbf{x}, \mathbf{v}, u_i]H^{(i)T} = \mathbf{0})$ admits a factorized form, computing $w^*(\mathbf{p}, u_i)$ simplifies to computing the marginal of a factorized function and finding its minimum value [60]. This allows for the construction of a graphical representation of the factorization. In particular, our aim is to compute $w^*(\mathbf{p}, u_i = 0)$ and $w^*(\mathbf{p}, u_i = 1)$.

Example 2.2.3. *Let us consider the case of u_2.*

In this particular case, $H^{(2)}$ is expressed as:

$$H^{(2)} = \begin{array}{c} \begin{matrix} x_0 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & v_0 & v_1 & u_2 \end{matrix} \\ \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \end{array} \tag{2.7}$$

we will denote by $x_i = p_i \oplus s_i$. *Equation 2.5 implies:*

$$[\mathbf{x}, \mathbf{v}, u_2]H^{(2)T} = 0 \Leftrightarrow \begin{cases} x_3 + x_7 + v_0 = 0 \\ x_1 + x_5 + v_0 = 0 \\ x_0 + x_4 + v_1 = 0 \\ x_2 + x_6 + v_1 = 0 \\ v_0 + v_1 + u_2 = 0 \end{cases} \tag{2.8}$$

Therefore:

$$\begin{aligned} \mathbb{1}([\mathbf{x}, \mathbf{v}, u_2]H^{(2)T} = 0) = \mathbb{1}(x_3 + x_7 + v_0 = 0)\mathbb{1}(x_1 + x_5 + v_0 = 0) \\ \mathbb{1}(x_0 + x_4 + v_1 = 0)\mathbb{1}(x_2 + x_6 + v_1 = 0) \\ \mathbb{1}(v_0 + v_1 + u_2 = 0) \end{aligned} \tag{2.9}$$

The factorised form of $\mathbb{1}([\mathbf{x}, \mathbf{v}, u_2]H^{(2)T} = 0)$ *ensures an efficient computation using factor graphs.*

The computation of $w^*(\mathbf{p}, 0)$ and $w^*(\mathbf{p}, 1)$ can be efficiently done using the message passing rules on factor graphs.

Two message passing configurations can be encountered. The first configuration is referred to as the parity node case (see Figure 2.2), where two variable nodes x_0 and x_1 are connected to a parity node x_2 through a parity function f.

The corresponding parity matrix H for the factor graph is also illustrated in the same figure. To each message coming from variable node to a parity node, we associate a vector $\boldsymbol{\mu}_{x_i \to f}$ defined as:

$$\boldsymbol{\mu}_{x_i \to f} = \begin{pmatrix} \mu^{(0)}_{x_i \to f} \\ \mu^{(1)}_{x_i \to f} \end{pmatrix} = \begin{pmatrix} w^*(T_{x_i}|x_i = 0)(X) \\ w^*(T_{x_i}|x_i = 1)(X) \end{pmatrix} \tag{2.10}$$

where T_{x_i} is defined as $T_{x_i} = \left\{\mathbf{x}| \quad \mathbf{x}H^T_{T_{x_i}} = 0\right\}$ and $H^T_{T_{x_i}}$ represents the parity matrix of each sub-graph associated to a variable node x_i.

As depicted in Figure 2.2, $\boldsymbol{\mu}_{f \to x_2}$ can be computed from $\boldsymbol{\mu}_{x_0 \to f}$ and $\boldsymbol{\mu}_{x_1 \to f}$ for $\{x_0, x_1, x_2\} \in \{0, 1\}^3$ using the message passing rules described in [60] for min-sum algorithm as follows:

$$\mu^{(x_2)}_{f \to x_2} = \min_{\{x_0, x_1\} \in \{0,1\}^2} (-\log(\mathbb{1}_{x_0 \oplus x_1 \oplus x_2 = 0}) + \mu^{(x_0)}_{x_0 \to f} + \mu^{(x_1)}_{x_1 \to f}) \tag{2.11}$$

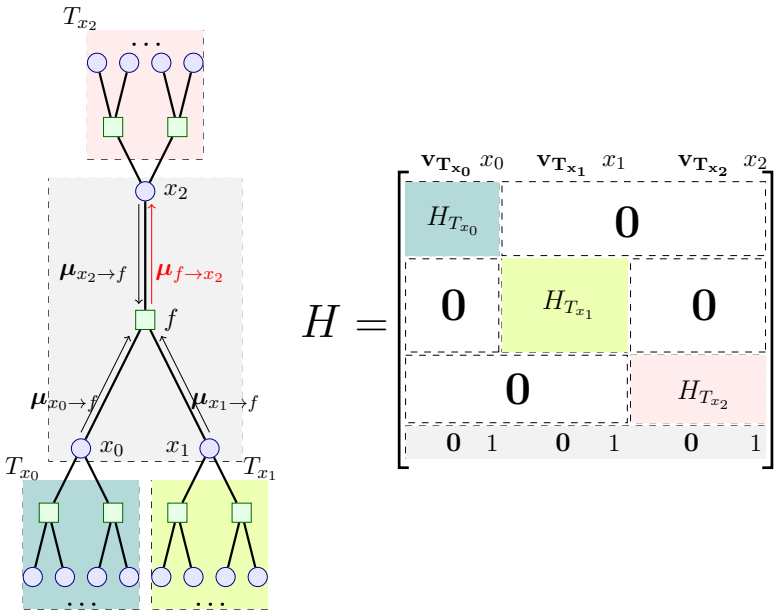


Figure 2.2: Parity node case

$\mu^{(0)}_{f\to x_2}$ and $\mu^{(1)}_{f\to x_2}$ can therefore be expressed as:

$$\begin{cases} \mu^{(0)}_{f\to x_2} = \min(\mu^{(0)}_{x_0\to f} + \mu^{(0)}_{x_1\to f}, \mu^{(1)}_{x_0\to f} + \mu^{(1)}_{x_1\to f}) \\ \mu^{(1)}_{f\to x_2} = \min(\mu^{(0)}_{x_0\to f} + \mu^{(1)}_{x_1\to f}, \mu^{(1)}_{x_0\to f} + \mu^{(0)}_{x_1\to f}) \end{cases} \tag{2.12}$$

The second configuration is referred to as the variable node case (see Figure 2.3). In this case, two parity nodes with parity function f_0 and f_1 are connected to a variable node x. The parity matrix associated to the graph is also given in the same figure. To each message coming from a parity node f_i, we associate:

$$\boldsymbol{\mu}_{f_i\to x} = \begin{pmatrix} \mu^{(0)}_{f_i\to x} \\ \mu^{(1)}_{f_i\to x} \end{pmatrix} = \begin{pmatrix} w^*(T_{f_i}|x=0)(X) \\ w^*(T_{f_i}|x=1)(X) \end{pmatrix} \tag{2.13}$$

Given the two incoming messages from the parity nodes to the variable node x, the messages $\mu^{(b)}_{x\to f_2}$, $b = \{0,1\}$, can be expressed as follows using the message passing rule:

$$\mu^{(b)}_{x\to f_2} = \mu^{(b)}_{f_0\to x} + \mu^{(b)}_{f_1\to x} \tag{2.14}$$

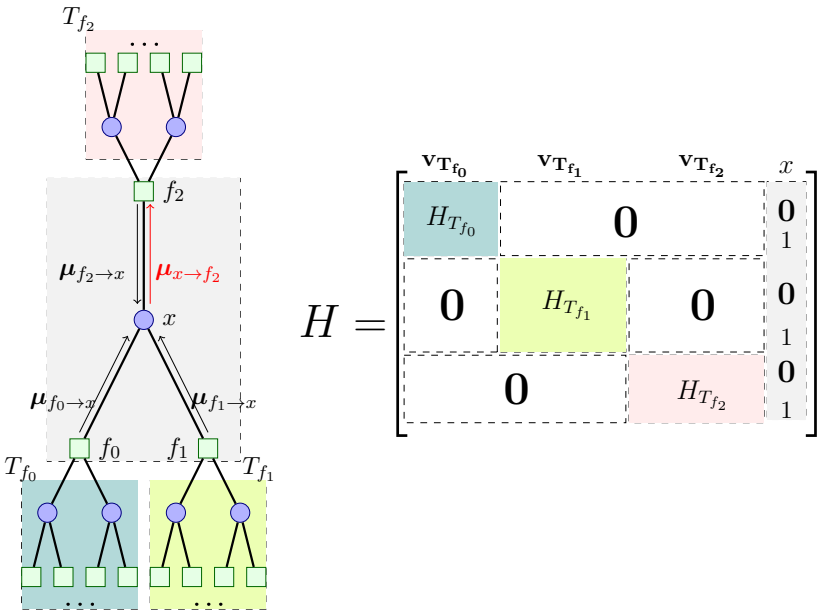


Figure 2.3: Variable node case

Example 2.2.4. *The Tanner Graph for decoding u_2 is given in Figure 2.4. It is an illustration of a decoding factor graph of bit u_2 for a polar code of length $N = 8$ with* $\mathbf{p} = [1, 0, 0, 0, 0, 0, 0, 0]$.

Using Equation (2.6) *and reference [60], the initial message that are sent from the leaf node that represents the minimum weight of each x_i can be expressed as:*

$$\boldsymbol{\mu}_{x_i} = \begin{pmatrix} p_i \oplus 0 \\ p_i \oplus 1 \end{pmatrix} \tag{2.15}$$

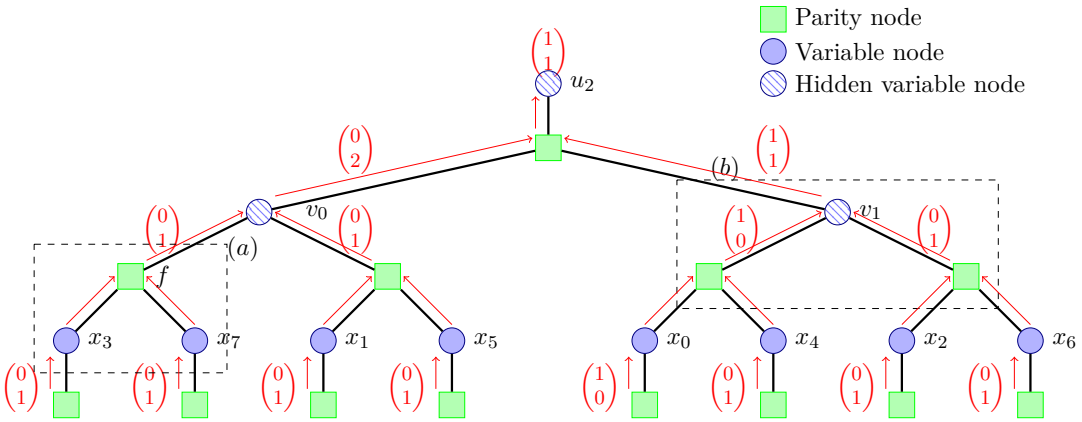


Figure 2.4: Tanner Graph of u_2 decoding for a polar code with $N = 8$

In this example, considering the dashed sub-graph (a) and using Equation (2.12)*, we obtain:*

$$\boldsymbol{\mu}_{f\to v_0} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{2.16}$$

2.2.2.2 Weight Enumeration Functions of polar cosets

In this section, we take advantage of the results introduced in [50] in order to simultaneously compute the minimum weight and its associated number of occurrences.
In [50], the Weight Enumeration Function (WEF) of a polar coset $\mathcal{C}_N(\mathbf{u}_0^i)$ is defined as:

$$A_N(\mathcal{C}_N(\mathbf{u}_0^i))(X) \triangleq \sum_{w=0}^{N} A_w X^w \tag{2.17}$$

where A_w denotes the number of vectors in $\mathcal{C}_N(\mathbf{u}_0^i)$ having a weight equal to w. It has been proven in [50] that as for the relations for bit channels for the LLR computation (Equations (23) and (24) of [1]), the weight enumeration function of polar cosets can be computed efficiently using different propagation rules in the case of parity nodes and check nodes. Again, during message passing formalism, two configurations can be encountered. The same configurations in Figures 2.2 and 2.3 are explored but this time with messages concerning the WEF.
The first configuration depicted in Figure 2.2 shows two variable nodes, x_0 and x_1, connected to a parity node x_2 via a parity function f.

To each message coming from a variable node, we associate:

$$\boldsymbol{\theta}_{x_i\to f} = \begin{pmatrix} \theta^{(0)}_{x_i\to f} \\ \theta^{(1)}_{x_i\to f} \end{pmatrix} = \begin{pmatrix} A_N(T_{x_i}|x_i=0)(X) \\ A_N(T_{x_i}|x_i=1)(X) \end{pmatrix} \tag{2.18}$$

In this case, following Equation (1) in [50], $\boldsymbol{\theta}_{f\to x_2}$ can be computed from $\boldsymbol{\theta}_{x_0\to f}$ and $\boldsymbol{\theta}_{x_1\to f}$ as:

$$\boldsymbol{\theta}_{f\to x_2} = \begin{pmatrix} \theta^{(0)}_{x_0\to f}\theta^{(0)}_{x_1\to f} + \theta^{(1)}_{x_0\to f}\theta^{(1)}_{x_1\to f} \\ (\theta^{(0)}_{x_0\to f}\theta^{(1)}_{x_1\to f} + \theta^{(1)}_{x_0\to f}\theta^{(0)}_{x_1\to f} \end{pmatrix} \tag{2.19}$$

Silimarly, the second configuration depicted in Figure 2.3 is where two parity nodes with parity functions f_0 and f_1 are connected to a variable node x. To each message coming

from a parity node f_i to a variable node x, we associate:

$$\boldsymbol{\theta}_{f_i \to x} = \begin{pmatrix} \theta^{(0)}_{f_i \to x} \\ \theta^{(1)}_{f_i \to x} \end{pmatrix} = \begin{pmatrix} A_N(T_{f_i}|x=0)(X) \\ A_N(T_{f_i}|x=1)(X) \end{pmatrix} \tag{2.20}$$

Given the two incoming messages $\boldsymbol{\theta}_{f_0 \to x}$ and $\boldsymbol{\theta}_{f_1 \to x}$ from the parity nodes to the variable node x, $\boldsymbol{\theta}_{x \to f_2}$ can be expressed using Equation (2) of [50] as:

$$\boldsymbol{\theta}_{x \to f_2} = \begin{pmatrix} \theta^{(0)}_{f_0 \to x} \theta^{(0)}_{f_1 \to x} \\ \theta^{(1)}_{f_0 \to x} \theta^{(1)}_{f_1 \to x} \end{pmatrix} \tag{2.21}$$

We introduce a Minimum Weight Enumeration Function (MWEF) of a polar coset $\mathcal{C}_N(\mathbf{u}_0^i)$ defined as:

$$A^*_N(\mathcal{C}_N(\mathbf{u}_0^i))(X) = A_{w^*} X^{w^*} \tag{2.22}$$

Where w^* denotes the minimum weight of the coset and A_{w^*} the number of vectors with minimum weight.
It is also possible to compute the MWEF of a polar coset while adapting the message passing equations for the WEF computation. In the first configuration, to each message coming from a variable node, we associate:

$$\boldsymbol{\theta}^*_{x_i \to f} = \begin{pmatrix} \theta^{*(0)}_{x_i \to f} \\ \theta^{*(1)}_{x_i \to f} \end{pmatrix} = \begin{pmatrix} A^*_N(T_{x_i}|x_i=0)(X) \\ A^*_N(T_{x_i}|x_i=1)(X) \end{pmatrix} \tag{2.23}$$

In that case, given the same configuration as in Figure 2.2, $\boldsymbol{\theta}^*_{f \to x_2}$ can be computed from $\boldsymbol{\theta}^*_{x_0 \to f}$ and $\boldsymbol{\theta}^*_{x_1 \to f}$ as:

$$\boldsymbol{\theta}^*_{f \to x_2} = \begin{pmatrix} LP\left(\theta^{*(0)}_{x_0 \to f} \theta^{*(0)}_{x_1 \to f} + \theta^{*(1)}_{x_0 \to f} \theta^{*(1)}_{x_1 \to f}\right) \\ LP\left(\theta^{*(0)}_{x_0 \to f} \theta^{*(1)}_{x_1 \to f} + \theta^{*(1)}_{x_0 \to f} \theta^{*(0)}_{x_1 \to f}\right) \end{pmatrix} \tag{2.24}$$

where $LP(.)$ denotes the operator that only selects the monomial of lower degree.
In the configuration described in Figure 2.3, to each message coming from a parity node f_i, we associate:

$$\boldsymbol{\theta}^*_{f_i \to x} = \begin{pmatrix} \theta^{*(0)}_{f_i \to x} \\ \theta^{*(1)}_{f_i \to x} \end{pmatrix} = \begin{pmatrix} A^*_N(T_{f_i}|x=0)(X) \\ A^*_N(T_{f_i}|x=1)(X) \end{pmatrix} \tag{2.25}$$

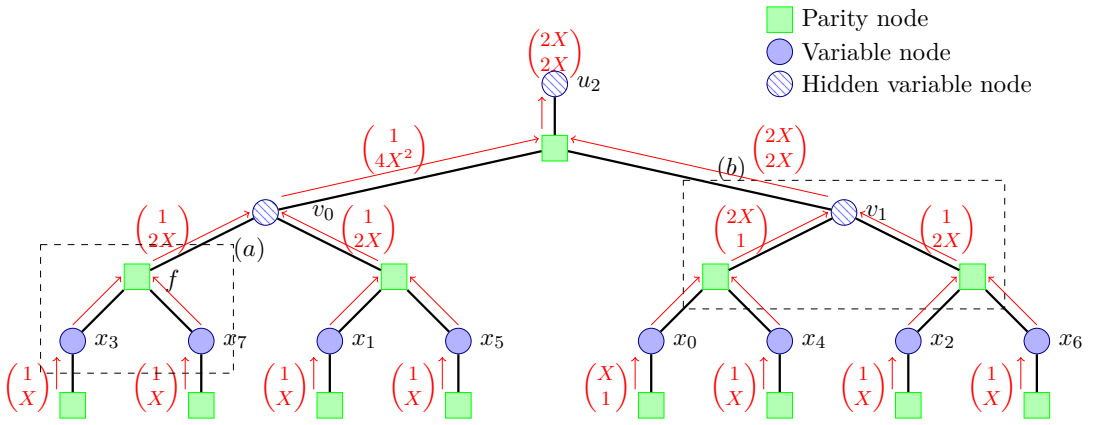


Figure 2.5: Tanner Graph of u_2 decoding for a polar code with $N = 8$

In that case,

$$\boldsymbol{\theta}^*_{x\to f_2} = \begin{pmatrix} \theta^{*(0)}_{f_0\to x} \theta^{*(0)}_{f_1\to x} \\ \theta^{*(1)}_{f_0\to x} \theta^{*(1)}_{f_1\to x} \end{pmatrix} \tag{2.26}$$

The proof of Equations (2.24) and 2.26 is given in Appendix A.

The initial message that are sent from the leaf node representing the MWEF of each x_i is:

$$\boldsymbol{\theta}_{x_i} = \begin{pmatrix} X^{(p_i \oplus 0)} \\ X^{(p_i \oplus 1)} \end{pmatrix} \tag{2.27}$$

Example 2.2.5. *Figure 2.5 gives an illustration of a decoding factor graph of bit u_2 for a polar code of length $N = 8$ with $\mathbf{p} = [1, 0, 0, 0, 0, 0, 0, 0]$.*

In this example, considering the dashed sub-graph (a), we have:

$$\boldsymbol{\theta}_{f\to v_0} = \begin{pmatrix} LP(1 + X^2) \\ LP(2X) \end{pmatrix} = \begin{pmatrix} 1 \\ 2X \end{pmatrix} \tag{2.28}$$

2.2.2.3 Extension to the reduced spectrum of polar cosets

In the previous section, we discussed the modification of the WEF computation to consider solely the minimum weight of polar cosets. Extending this computation, we can evaluate the reduced spectrum of polar cosets. This is obtained by introducing a Relaxed

Minimum Weight Enumeration Function (RWEF) for a polar coset, which characterizes the weight distribution of the polar coset for weights less than or equal to w_{end}. This RWEF $A_N^{w_{\text{end}}}(\mathcal{C}_N(\mathbf{u}_0^i))(X)$ of a polar coset $\mathcal{C}_N(\mathbf{u}_0^i)$ is defined as:

$$A_N^{w_{\text{end}}}(\mathcal{C}_N(\mathbf{u}_0^i))(X) = \sum_{w=0}^{w_{\text{end}}} A_w X^w \tag{2.29}$$

As for the MWEF, it is possible to compute the RWEF using the following message passing relations:

- In the parity node case depicted in Figure 2.2, we define $\boldsymbol{\theta}_{x_i \to f}^{w_{end}}$ as:

$$\boldsymbol{\theta}_{x_i \to f}^{w_{end}} = \begin{pmatrix} \theta_{x_i \to f}^{w_{end}(0)} \\ \theta_{x_i \to f}^{w_{end}(1)} \end{pmatrix} = \begin{pmatrix} A_N^{w_{end}}(T_{x_i}|x_i = 0)(X) \\ A_N^{w_{end}}(T_{x_i}|x_i = 1)(X) \end{pmatrix} \tag{2.30}$$

  In this case, $\boldsymbol{\theta}_{f \to x_2}^{w_{end}}$ can be computed from $\boldsymbol{\theta}_{x_0 \to f}^{w_{end}}$ and $\boldsymbol{\theta}_{x_1 \to f}^{w_{end}}$ as:

$$\boldsymbol{\theta}_{f \to x_2}^{w_{end}} = \begin{pmatrix} LT_{w_{end}}\left(\theta_{x_0 \to f}^{w_{end}(0)} \theta_{x_1 \to f}^{w_{end}(0)} + \theta_{x_0 \to f}^{w_{end}(1)} \theta_{x_1 \to f}^{w_{end}(1)}\right) \\ LT_{w_{end}}\left(\theta_{x_0 \to f}^{w_{end}(0)} \theta_{x_1 \to f}^{w_{end}(1)} + \theta_{x_0 \to f}^{w_{end}(1)} \theta_{x_1 \to f}^{w_{end}(0)}\right) \end{pmatrix} \tag{2.31}$$

  where $LT_{w_{\text{end}}}(.)$ denotes the operator that only selects the monomials of a degree lower or equal to w_{end}.

- Similarly, in the configuration described in Figure 2.3, to each message coming from a parity node f_i, we associate $\boldsymbol{\theta}_{f_i \to x}^{w_{end}}$ defined as:

$$\boldsymbol{\theta}_{f_i \to x}^{w_{end}} = \begin{pmatrix} \theta_{f_i \to x}^{w_{end}(0)} \\ \theta_{f_i \to x}^{w_{end}(1)} \end{pmatrix} = \begin{pmatrix} A_N^{w_{end}}(T_{f_i}|x = 0)(X) \\ A_N^{w_{end}}(T_{f_i}|x = 1)(X) \end{pmatrix} \tag{2.32}$$

  In that case:

$$\boldsymbol{\theta}_{x \to f_2}^{w_{end}} = \begin{pmatrix} LT_{w_{end}}\left(\theta_{f_0 \to x}^{w_{end}(0)} \theta_{f_1 \to x}^{w_{end}(0)}\right) \\ LT_{w_{end}}\left(\theta_{f_0 \to x}^{w_{end}(1)} \theta_{f_1 \to x}^{w_{end}(1)}\right) \end{pmatrix} \tag{2.33}$$

The proof for equations (2.31) and (2.33) is given in Appendix B.

Example 2.2.6. *Figure 2.6 gives an illustration of the computation of the RWEFs* $A_8^3(\mathcal{C}_8([1,0],0))$ *and* $A_8^3(\mathcal{C}_8([1,0],1))$ *for a polar code of length* $N = 8$. *In this case,* $\mathbf{p} = [1,0,0,0,0,0,0,0]$ *and* $w_{end} = 3$. *The computation is performed bu applying Equation* (2.31) *in the case of a parity node and Equation* (2.33) *in the case of a check node. We*

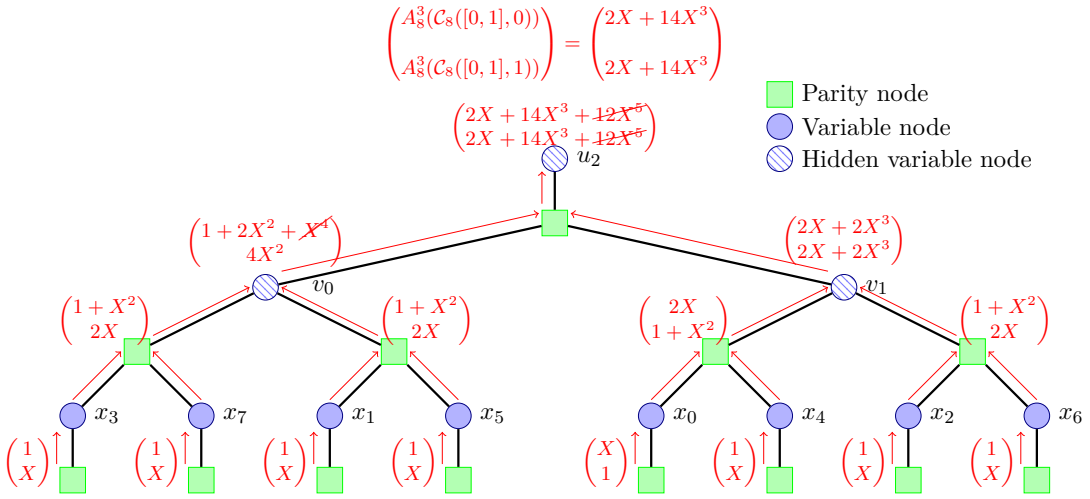


Figure 2.6: Factor graph of u_2 for the computation of $A_8^3(\mathcal{C}_8([1,0],0))$ and $A_8^3(\mathcal{C}_8([1,0],1))$

can see from the figure that at each computation step, the monomials with a degree greater than 3 *are eliminated.*

2.2.3 Computational complexity analysis

In this section, we further explore the computational complexity for the computation of the MWEF, and RWEF of polar cosets.

- For the computation of the MWEF, depending on the configuration (parity node case or variable node case), a certain number of arithmetic operations has to be performed.
  In the parity node case, equation (2.24) is applied. This results in at most 12 arithmetic operations. In the variable node case, Computation of (2.26) results in 4 arithmetic operations (two multiplications and two additions). Given that on each graph factor of u_i, $N-1$ nodes have to be evaluated, the computational complexity defined as the total number of arithmetic operations is at most equal to $12(N-1)$. Moreover, is is important to note that the computation of the minimum weight for polar cosets only necessitates $6(N-1)$ arithmetic operations, employing the same logic.

- For the computation of the RWEF up to a certain weight w_{end}, the polynomial $A_N^{w_{\text{end}}}(X)$ is comprised of a maximum number of monomials equal to w_{end}. In the following, we will focus on this worst case.
  As for the computation of the MWEF, depending on the configuration a certain

number of arithmetic operations has to be performed.
In the parity node, the execution of Equation (2.31) results in $8w_{\text{end}}^2$ arithmetic operations. In the variable node case, Equation (2.33) requires $4w_{\text{end}}^2$ arithmetic operations. The parity node case has the highest complexity, so we will only consider this case.
Given that on each factor graph of u_i, $N-1$ nodes have to be evaluated, the overall computational complexity is at most equal to $8(N-1)w_{\text{end}}^2$. It's important to note that the worst-case computational complexity of the RWEF is described by considering a scenario that is highly unlikely in practice. Actually, the number of elements in the RWEF up to a weight w_{end} is significantly less than w_{end}.

2.3 Low complexity algorithm for the computation of minimum distance properties for pure and pre-transformed polar codes

2.3.1 Pure polar code case

We proved in Section 2.2 that we are able to compute the minimum weight and its related number of occurrences for any polar coset $\mathcal{C}_N(\mathbf{u}_0^i)$. In this section, we express any polar code as a union of disjoint cosets and take advantage of the results of Section 2.2 to compute the overall number of occurrences of codewords with minimum weight for a polar code.

As described in [50], any polar code $\mathcal{C}$ can be formulated as the disjoint union of the following cosets:

$$\mathcal{C} = \bigcup_{\mathbf{u}_0^{s-1} \in L_s} \mathcal{C}_N(\mathbf{u}_0^{s-1}, u_s = 0) \tag{2.34}$$

where s denotes the index of the last frozen bit and the set

$$L_s = \left\{ \mathbf{u}_0^{s-1} \in \{0,1\}^s | u_i = 0, \forall i \in \mathcal{F} \right\}$$

L_s represents the set of all the possible prefixes while taking into account the frozen bits.

The expression given in Equation (2.34) is explained by the fact that after the last frozen bit s, the suffixes $\mathbf{u}_{s+1}^{N-1}$ take values in $\{0,1\}^{N-1-s}$ as they are all information bits. These

suffixes, along with their associated prefixes, collectively describe the entirety of the code. In order to compute the overall minimum distance and the corresponding number of occurrences, one approach is to compute the minimum weight of each coset comprising $\mathcal{C}$ separately. Then, by tallying the occurrences of codewords with the overall minimum weight, the number of codewords with minimum weight is obtained based on Equation (2.34). This can be done by computing the MWEF of each coset composing $\mathcal{C}$.

An issue arises when examining the coset $\mathcal{C}_N(\mathbf{u}_0^s)$, as its minimum weight is equal to 0 because of the presence of the all-zero codeword. Consequently, there is a risk of missing potential codewords with minimum weight that might be found within $\mathcal{C}_N(\mathbf{u}_0^s)$ if we solely focus on computing its minimum weight. The proposed solution is to compute for the coset $\mathcal{C}_N(\mathbf{u}_0^s)$ the Reduced Weight Enumeration Function with $w_{\text{end}} = d^*(\mathcal{C})$ instead of the Minimum Weight Enumeration Function. To this matter, we express $A^*_{\mathcal{C}\setminus\{\mathbf{0}\}}(X)$ the MWEF of $\mathcal{C} \setminus \{\mathbf{0}\}$ as:

$$A^*_{\mathcal{C}\setminus\{\mathbf{0}\}}(X) = \text{LP}\left(\sum_{\substack{\mathbf{u}_0^s \in L_s \\ \mathbf{u}_0^s \neq \mathbf{0}_0^s}} A^*_N(\mathcal{C}_N(\mathbf{u}_0^s))(X) + A^{d^*}_N(\mathcal{C}_N(\mathbf{0}_0^s)(X)) - 1\right) \tag{2.35}$$

Equation (2.35) results from exploring the minimum weight and its associated number of occurrences for all the remaining cosets in the list at the s^{th} step, except for the coset $\mathcal{C}_N(\mathbf{u}_0^s)$ where the RWEF is evaluated for $w_{end} = d^*$. Subsequently, the minimum distance is derived from the exponent of the monomial $A^*_N(\mathcal{C}_N(\mathbf{u}_0^s))$, while the number of occurrences is determined by its coefficient.

In order to achieve a deterministic computation of d^* and A^*, we need to compute the MWEF for each coset included in $\left\{\mathcal{C} \setminus \mathbf{0}_0^{N-1}\right\}$. The total number of cosets to explore can therefore be expressed as:

$$n_c = (2^\gamma - 1) + (N - s) \tag{2.36}$$

where γ, referred to as the mixing factor in [50], denotes the total number of information bits before the last frozen bit.
It is clear that even for moderate code sizes, the resulting computational complexity may become prohibitive. For this reason, we propose an algorithm that provides a deterministic computation of both the minimum distance and its number of occurrences while only exploring the relevant cosets for the computation of d^* and A^*.

2.3 Low complexity algorithm for the computation of minimum distance properties for pure and pre-transformed polar codes

Knowing that we are able to compute the MWEF for any polar coset $\mathcal{C}_N(\mathbf{u}_0^i)$, it is possible to propose an enumeration structure similar to a list decoder that has the advantage of pruning cosets with a constraint of their minimal weight and only exploring relevant cosets.

We note that in the case of pure polar codes, the minimum distance $d^*(\mathcal{C})$ of a code $\mathcal{C}$ is known as explained in Chapter 1. Therefore the known value of the overall code minimum distance can be used as a threshold to eliminate irrelevant paths. Algorithm 1 gives the details of the proposed approach. It operates as follows:

1. For each $i \in [\![0, s]\!]$, all the possible information vectors $\mathbf{u}_0^i$ at an exploration step i are listed.

2. For each of the aforementioned paths, $w^*(\mathcal{C}_N(\mathbf{u}_0^{i-1}, u_i))$ is computed.

3. The cosets with $w^*(\mathcal{C}_N(\mathbf{u}_0^{i-1}, u_i)) > d^*(\mathcal{C})$ are discarded. Those cosets are irrelevant to the computation of the total number of occurrences as equation (2.2) leads to:

$$w^*(\mathcal{C}_N(\mathbf{u}_0^{N-1})) \leqslant w^*(\mathcal{C}_N(\mathbf{u}_0^1)) \leqslant ... \leqslant w^*(\mathcal{C}_N(u_0)) \tag{2.37}$$

   This means that $\forall \quad \mathcal{C}_N(\mathbf{u}_0^{i+1})$ such that $w^*(\mathcal{C}_N(\mathbf{u}_0^{i+1})) < d^*(\mathcal{C})$, $\nexists j \in [\![i+1; N-1]\!]$ such that $w^*(\mathcal{C}_N(\mathbf{u}_0^{j+1})) = d^*(\mathcal{C})$. In other words, no codeword with minimum weight can be found in a coset whose minimum weight is greater than $d^*(\mathcal{C})$.

4. When $i = s$, the MWEF is computed for all the cosets remaining in the list except for the coset $\mathcal{C}_N(\mathbf{0}_0^s)$. The number of cosets with minimum weight is counted.

5. For the coset $\mathcal{C}_N(\mathbf{0}_0^s)$, the RWEF with $w_{end} = d^*(\mathcal{C})$ is computed and the overall number of occurrences $A^*(\mathcal{C})$ of the polar code is obtained thanks to Equation (2.35).

Figure 2.7 gives an insight of Algorithm 1 for a $(32, 10)$ polar code. In this case, $d^*(\mathcal{C}) = 8$, $s = 25$ and the information set $\mathcal{I} = \{15, 22, 23, 24, 26, 27, 28, 29, 30, 31\}$. The information bits are represented in green and the frozen bits in red. At each enumeration stage, a path that is eliminated for having a minimum weight greater that the code's overall minimum distance is represented in gray. It means that at each decoding step, any coset that has a minimum weight greater than 8 is discarded. When reaching the last frozen bit, the MWEF is computed for each one of the remaining paths and the overall MWEF of the code is computed.

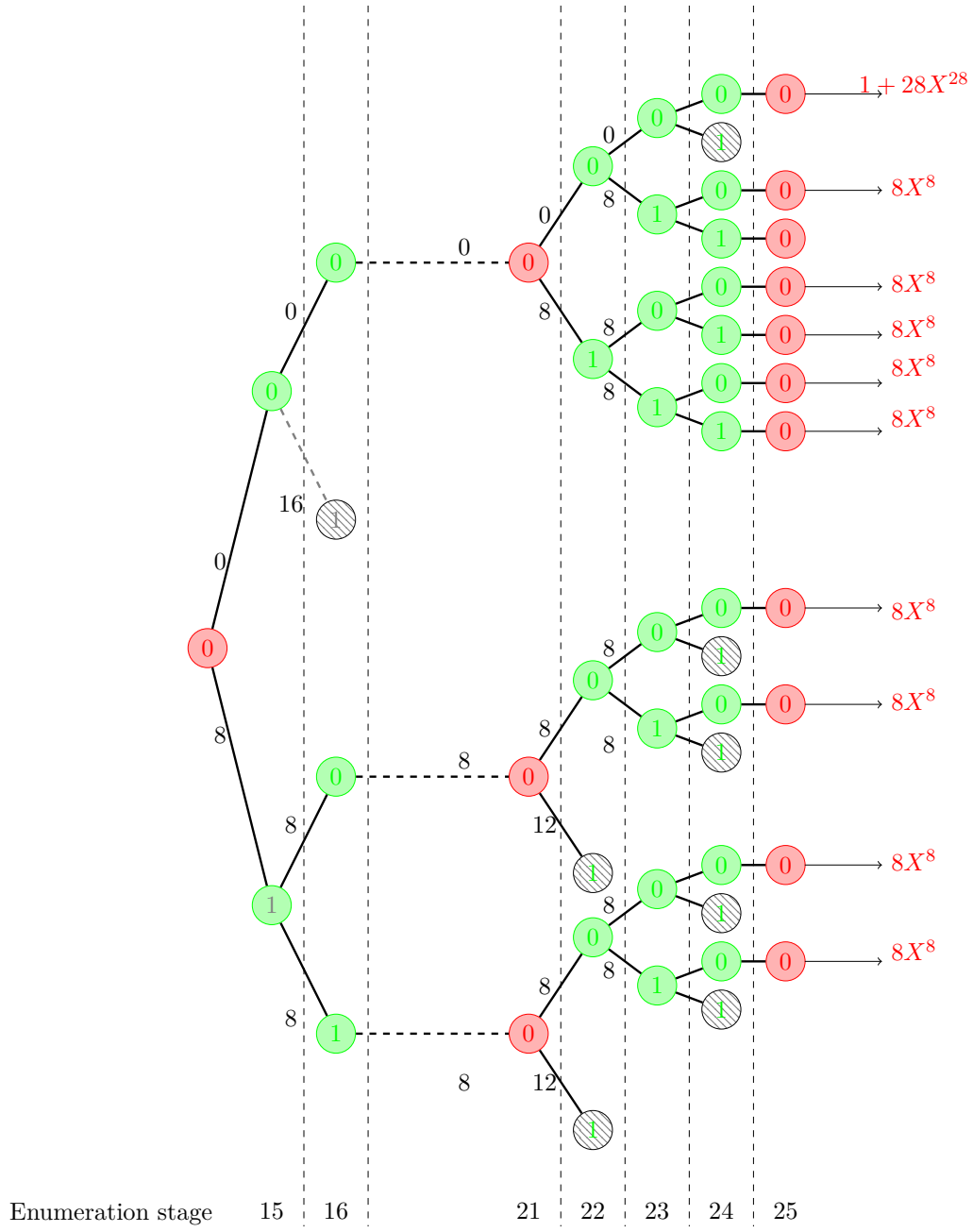


Figure 2.7: Example of Algorithm 1 for a $(32, 10)$ polar code

Algorithm 1: Computing $A^*(\mathcal{C})$ of a pure polar code

Input: Polar code $\mathcal{C}(N, K, \mathcal{F})$

$s \leftarrow$ index of the last frozen bit
$L \leftarrow 1$
$\mathcal{L} \leftarrow \{0\}$ `/* List to store prefixes */`

for $i \in [\![0; s]\!]$ **do**
  for $l \in [\![1; L]\!]$ **do**
    Compute w^* of the cosets in the list
    Discard the cosets for which $w^* > d^*(\mathcal{C})$
    $L \leftarrow |\mathcal{L}|$
  end
  if $i = s$ **then**
    Compute the MWEF of the remaining cosets in the list except for $\mathcal{C}_N(\mathbf{0}_0^s)$
    Compute $A_N^{d^*(\mathcal{C})}(\mathcal{C}_N(\mathbf{0}_0^s))(X)$
    $A^* \leftarrow$ Sum of the computed occurrences
  end
end
Return $(A^*(\mathcal{C}))$

2.3.2 Results

Let C_i denote the number of remaining cosets at an enumeration step i. The total number of evaluated cosets n_r is:

$$n_r = \sum_{i=0}^{s} C_i \tag{2.38}$$

Algorithm 1 consists in $s-1$ loop iterations where the minimum weight of C_i cosets is evaluated, and one loop (When $i = s$) where the MWEF of the remaining cosets in the list is evaluated except for the coset $\mathcal{C}(\mathbf{u}_0^s)$ where the RWEF with $w_{end} = d^*(\mathcal{C})$ is evaluated. As the computational complexity for computing the minimum weight, MWEF and RWEF up to $w_{end} = d^*(\mathcal{C})$ of a polar coset are at most equal to $6(N-1)$, $12(N-1)$ and $8d^*(\mathcal{C})^2(N-1)$, the overall computational complexity TC^* of Algorithm 1 is at most equal to:

$$TC^* = \sum_{i=0}^{s-1} 6(N-1)C_i + 12(N-1)(C_s - 1) + 8d^*(\mathcal{C})^2(N-1) \tag{2.39}$$

Algorithm 1 has been applied on a wide range of polar codes, for different code rates and

N	128		256		512		1024		2048	
K	96	64	192	128	384	256	768	512	1536	1024
R	$\frac{3}{4}$	$\frac{1}{2}$	$\frac{3}{4}$	$\frac{1}{2}$	$\frac{3}{4}$	$\frac{1}{2}$	$\frac{3}{4}$	$\frac{1}{2}$	$\frac{3}{4}$	$\frac{1}{2}$
s	73	98	193	201	385	417	771	897	1545	1809
γ	41	34	129	73	257	161	515	385	1033	785
d^*	4	8	8	8	8	8	8	16	8	16
A^*	96	304	61536	96	53440	64	24960	36032	11008	896
n_c	$2.1e^{12}$	$1.7e^{10}$	$6.8e^{38}$	$9.4e^{21}$	$2.3e^{77}$	$2.9e^{48}$	$1.07e^{155}$	$7.8e^{115}$	INF	$2e^{236}$
n_r	114	167	406055	210	317031	450	358938	178870	43489	1555
C_{max}	1	10	15122	1	6690	1	2113	4514	385	1

TABLE 2.3: Minimum distance and number of occurrences values for different polar codes

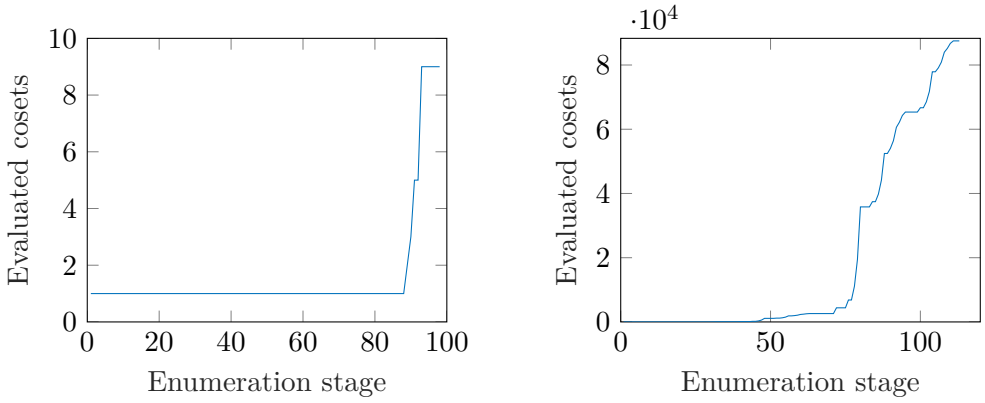
a code length up to $N = 2048$ [2]. For the polar codes with $N \leq 1024$, the frozen bits set is the one specified in the 5G standard [9]. For $N = 2048$, the frozen bits set is generated with density evolution [62] with design $E_b/N_0 = 3dB$.

Table 2.3 summarizes the minimum distance d^*, the associated occurrences A^*, the last frozen bit s, the mixing factor γ, the total number of explored cosets n_r and the maximum list size reached $C_{\max} = \max_{i \in [\![0, N-1]\!]} C_i$ for different polar codes.

It is important to point out that the simplification introduced in Algorithm 1 drastically reduces the number of explored cosets. This can be observed by comparing n_c and n_r. The reduction in terms of complexity is significant, which explains the ability of the proposed algorithm to easily process any type of polar code.

Figure 2.8 shows the evolution of the number of explored coset at each enumeration step until the last frozen bit is reached for a $(128, 64)$ polar code under 5G and RM constructions. We can see from this figure that in the case of polar code under 5G construction represented in Figure 2.8a, a maximum number of 9 cosets is explored, whereas the maximum number of explored cosets is equal to 87509 in the case of a RM construction. The $(128, 64)$ polar code constructed under 5G standard has parameters $d^* = 8$ and $A^* = 304$, whereas it has parameters $d^* = 16$ and $A^* = 94488$ under RM construction. This shows that the number of explored cosets varies following the considered code construction and is higher for codes with a higher number of codewords with minimum weight.

[2] We can even compute d^* and A^* for larger code sizes and for any desired code rate.



(a) (128,64) polar code under 5G construction

(b) (128,64) polar code under RM construction

Figure 2.8: Number of evaluated cosets at each enumeration step

2.3.3 Extension to pre-transformed polar codes

2.3.3.1 Representation of pre-transformed polar codes with polar cosets

We show in this section that the previous work on polar codes can be extented to PAC codes and polar codes with dynamic frozen bits. To start with, it is interesting to observe that as the pre-transformation for polar codes with dynamic frozen bits is only applied to frozen bits, they can be seen like PAC codes where the generator function $\mathbf{g}$ is dynamic following the decoding step i and each generator function $\mathbf{g}_i$ at a decoding step i can be expressed as:

$$\mathbf{g}_i = \begin{cases} [g_0, g_1, ..., g_{i-1}] & \text{if} \quad i \in \mathcal{F} \\ [1] & \text{otherwise} \end{cases} \tag{2.40}$$

Given this observation, for the rest of this section, we will consider only PAC codes, as the difference between PAC codes and Polar codes with dynamic frozen bits lies solely in their precoding rules.
Similar to regular polar codes, a PAC code $\mathcal{C}_{PAC}$ can be defined as follows:

$$\mathcal{C}_{\text{PAC}} = \bigcup_{\mathbf{u}_0^s \in \mathcal{B}_s} \mathcal{C}_N(\mathbf{u}_0^{s-1}, u_s) \tag{2.41}$$

$$\begin{array}{r} u_0 = 0 \\ u_1 = 0 \\ u_2 = 0 \\ u_3 = 0 \\ u_4 = 0 \\ u_5 = 0 \\ u_6 = v_6 \\ u_7 = v_7 + v_5 \\ u_8 = v_8 + v_6 \\ u_9 = v_9 + v_7 \\ u_{10} \\ u_{11} \\ u_{12} \\ u_{13} \\ u_{14} \\ u_{15} \end{array} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Figure 2.9: PAC transformation with $N = 16$ and $K = 8$

where $\mathcal{B}_s$ defines the set:

$$\mathcal{B}_s = \begin{cases} \mathbf{u}_0^s \in \{0,1\}^{s+1} | u_i = \mathbf{g}(\mathbf{v}_0^i), \forall i \in [\![0; s]\!] \\ v_i \in \{0,1\}, \forall i \notin \mathcal{F}, v_i = 0, \forall i \in \mathcal{F} \end{cases} \tag{2.42}$$

Example 2.3.1. *Figure 2.9 depicts a* $(16, 9, [1, 0, 1])$ *PAC code with a frozen index set* $\mathcal{F} = \{0, 1, 2, 3, 4, 5, 8, 9\}$.

As shown in figure 2.9, the red bits denote the information bits before the last frozen bit. The blue bits represent the information bits after the last frozen bit. The overall PAC code can be represented as:

$$\mathcal{C}_{PAC} = \bigcup_{\mathbf{u}_0^9 \in \mathcal{B}_9} \mathcal{C}_N(\mathbf{u}_0^8, u_9 = 0) \tag{2.43}$$

where $\mathcal{B}_9$ *defines the set:*

$$\mathcal{B}_9 = \left\{ \mathbf{u}_0^9 \in \{0,1\}^{10} | \left\{ \begin{array}{c} u_i = 0 \forall i \in [\![0; 5]\!] \\ u_i = v_i + v_{i-2} \forall i \in [\![6; 9]\!] \\ u_i = \{0,1\} \forall i \in [\![10; 15]\!] \\ v_i \in \{0,1\} \forall i \notin \mathcal{F}, v_i = 0 \forall i \in \mathcal{F} \end{array} \right. \right\}$$

2.3.3.2 Relaxed minimum distance properties of pre-transformed polar codes

To the best of the author's knowledge, there is no explicit way to directly compute the minimum distance of pre-transformed polar codes. In this section, we are focusing on the determination of both minimum distance and its number of occurrences for pre-transformed polar codes.

Similarly to the pure polar codes, the MWEF $A^*_{\mathcal{C}_{PAC}\setminus\{0\}}(X)$ can be expressed as:

$$A^*_{\mathcal{C}_{PAC}\setminus\{0\}}(X) = \mathrm{LP}\left(\sum_{\substack{\mathbf{u}_0^s \in \mathcal{B}_s \\ \mathbf{v}_0^s \neq \mathbf{0}_0^s}} A^*_N(\mathcal{C}_N(\mathbf{u}_0^s))(X) + A^{d^*}_N(\mathcal{C}_N(\mathbf{0}_0^s))(X) - 1\right) \tag{2.44}$$

The overall algorithm that describes the computation of d^* and A^* is given in Algorithm 2. There are two main differences compared to Algorithm 1:

- At each decoding step i, the remaining $\mathbf{v}_0^i$ are listed and a pre-transformation is applied to obtain the different $\mathbf{u}_0^i$. w^* is computed for each one of $\mathbf{u}_0^i$.
- There is no direct way to compute the minimum distance of PAC codes. As demonstrated in [25], any rate-1 linear pre-transformation on polar codes including convolutional encoding does not deteriorate the minimum distance properties of polar codes. This means that the minimum distance of PAC code $\mathcal{C}_{PAC}(N, K, \mathcal{F}, \mathbf{g})$ is greater or equal to the minimum distance of the polar code $\mathcal{C}(N, K, \mathcal{F})$. Therefore, an incremental method is used to determine d^*. We first set the threshold on the minimum weight of explored cosets to $d^*(\mathcal{C})$, i.e $w_{start} = d^*(\mathcal{C})$. If no codewords can be enumerated at the very last enumeration step, w_{start} is incremented until finding codewords whose minimum distance is equal to w_{start}. This achieves determining $d^*(\mathcal{C}_{PAC})$ and $A^*(\mathcal{C}_{PAC})$.

2.3.3.3 Minimum distance and associated number of occurrences results for pre-transformed polar codes

The analysis of the computational complexity of Algorithm 2 follows the same rules as the complexity of Algorithm 1.

Algorithm 2 was applied on a range of PAC codes, for different code rates and different

Algorithm 2: Computing d^* and A^* of a PAC code

Input: PAC code $\mathcal{C}_{PAC}(N, K, \mathcal{F}, \mathbf{g})$

$s \leftarrow$ index of the last frozen bit
$L \leftarrow 1$
$d_{start} \leftarrow d^*(\mathcal{C})$
$A^*(\mathcal{C}_{PAC}) = 0$
while $A^*(\mathcal{C}_{PAC}) = 0$ **do**
  for $i \in [\![1; s]\!]$ **do**
    if $i \in \mathcal{F}$ **then**
      for $l \in [\![1; L]\!]$ **do**
        $v_i[l] \leftarrow 0$
        $\hat{u}_i[l] = \mathbf{g}(\hat{\mathbf{v}}_0^i[l]))$
        Compute w^* and A^* of $\mathcal{C}_N(\mathbf{u}_0^{i-1}[l], u_i[l]))$
        Discard the paths for which $w^* > d^*$
        $L \leftarrow |\mathcal{L}|$
      end
    else
      $\mathcal{L} \leftarrow \mathcal{L} \bigcup \mathcal{L}'$ `/* 𝓛′ is a copy of 𝓛 */`

      for $l \in [\![1; L]\!]$ **do**
        $[v_i[l], v_i[l']] \leftarrow [0, 1]$
        $\hat{u}_i[l] = \mathbf{g}(\hat{\mathbf{v}}_0^i[l]))$
        $\hat{u}_i[l'] = \mathbf{g}(\hat{\mathbf{v}}_0^i[l']))$
        Compute w^* and A^* of $\mathcal{C}_N(\mathbf{u}_0^{i-1}[l], u_i[l]))$ and $\mathcal{C}_N(\mathbf{u}_0^{i-1}[l'], u_i[l']))$
        Discard the cosets for which $w^* > d^*$
        $L \leftarrow |\mathcal{L}|$
        if $i = s$ **then**
          Compute the MWEF of the remaining cosets in the list except for $\mathcal{C}_N(\mathbf{0}_0^s)$
          Compute $A_N^{d^*(\mathcal{C})}(\mathbf{0}_0^s)(X)$
          $A^* \leftarrow$ Sum of the computed occurrences
      end
    end
  end
  $A^* \leftarrow$ Occurrences of d^*
  Return (d^*, A^*)
  $d^* \leftarrow d^* + 2$
end

(N, K)	$\mathcal{F}$	d^*	A^*_{PAC}	C_{max}	n_r	$\sum_{i\in B} 2^{\lvert\mathcal{K}_i^f\rvert}(f-i)$ [58]
$(64, 22)$	RM	16	500	57	1061	65192
$(128, 64)$	RM	16	3120	2825	58329	7265295
$(256, 192)$	GA 4dB	8	53456	10326	227624	977664
$(512, 256)$	GA 2dB	16	36256	6074	139822	133968

TABLE 2.4: Minimum distance and number of occurrences

frozen set constructions. We will refer to the Gaussian Approximation (GA) construction with design-SNR of XdB [63] as GA XdB and to the Reed Muller construction as RM. Table 2.4 summarizes the minimum distance d^*, the associated occurrences A^* for both polar and PAC configurations. The (d^*, A^*) values highlighted in green were corroborated with results in [58]. It is possible to estimate the computational complexity of [58] as the total number of explored cosets : $\sum_{i\in B} 2^{\lvert\mathcal{K}_i^f\rvert}(f-i)$.

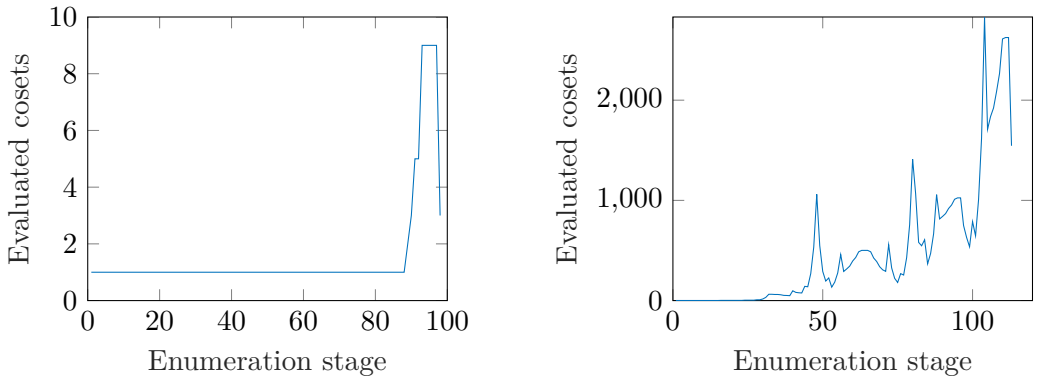
The complexity comparison also shows that the proposed algorithm clearly outperforms the algorithm proposed in [58]. In particular, for RM constructions, the number of explorations reduction is very significant. In terms of execution time, our MATLAB code time execution was compared to the MATLAB code time execution of [58] that can be found in [64] using a computer with 2 cores i5 and a 3.1GHz processor. For a (128,64) PAC code with a RM construction, our simulation time is around 11 seconds whereas [64]'s running time is around 25 minutes. Note that unlike Algorithm 1, the algorithm proposed in [58] is only valid for polar and RM constructions, no comparison can be undertaken with other kind of frozen set constructions.

Table 2.5 summarizes the values of d^* and A^* for codes with block length $N = \{128, 256, 512\}$, $R = \{1/3, 1/2, 3/4\}$ with the frozen bit sets specified in the 5G standard [9] and different generator polynomials. It can be seen from the table that the choice of the generator polynomial can have different impacts on the number of codewords with minimum weight.
In particular, in the case of $(128, 96)$, $(256, 128)$ and $(512, 256)$, the precoding have no impact on the number of codewords with minimum weight. This is due to the fact that all the codewords with minimum weights are located in the cosets $\mathcal{C}_N(\mathbf{0}_0^{i-1}, u_i = 1), i \in [s+1, N-1]$. As those cosets are the same for polar and PAC codes, the precoding has no impact on these particular cosets. The impact of the pre-transformation on the number of codewords with minimum weights has been further discussed in [65].

The evolution of the number of explored cosets at each enumeration step until reaching

N	g	$R = 1/3$			$R = 1/2$			$R = 3/4$		
		d^*	A^*	n_r	d^*	A^*	n_r	d^*	A^*	n_r
128	[1] (Polar)	16	1304	1826	8	304	167	4	96	114
	[1,0,1]	16	1304	1581	8	304	127	4	96	73
	[1,0,1,1,0,1,1]	16	776	711	8	256	121	4	96	58
	[1,0,1,1,0,1,1,0,1,1]	16	824	798	8	288	125	4	96	58
256	[1] (Polar)	16	816	788	8	96	210	8	61536	406055
	[1,0,1]	16	816	715	8	96	210	8	61536	390870
	[1,0,1,1,0,1,1]	16	688	390	8	96	210	8	36256	131814
	[1,0,1,1,0,1,1,0,1,1]	16	688	399	8	96	210	8	36488	133022
512	[1] (Polar)	16	352	654	8	64	450	8	53440	317031
	[1,0,1]	16	352	578	8	64	450	8	53440	317031
	[1,0,1,1,0,1,1]	16	224	526	8	64	450	8	40640	172639
	[1,0,1,1,0,1,1,0,1,1]	16	256	546	8	64	450	8	42688	187347

TABLE 2.5: Minimum distance and number of occurrences parameters for PAC codes



(a) (128,64) PAC code under 5G construction

(b) (128,64) PAC code under RM construction

Figure 2.10: Number of evaluated cosets at each enumeration step for a $(128, 64)$ PAC code

the last frozen bit for a $(128, 64)$ PAC code under 5G and RM constructions is shown in Figure 2.10. We can see from this figure that in the case of 2.10a, a maximum number of 9 cosets is explored, which have been reduced to 3 at the last enumeration step due to the precoding, whereas the maximum number of explored cosets is equal to 2625 in comparison to 87509 for a pure polar code in the case of a RM construction represented in Figure 2.8b. This proves that the precoding not only reduces the final number of occurrences but also reduces the number of cosets with minimum weight at each enumeration step.

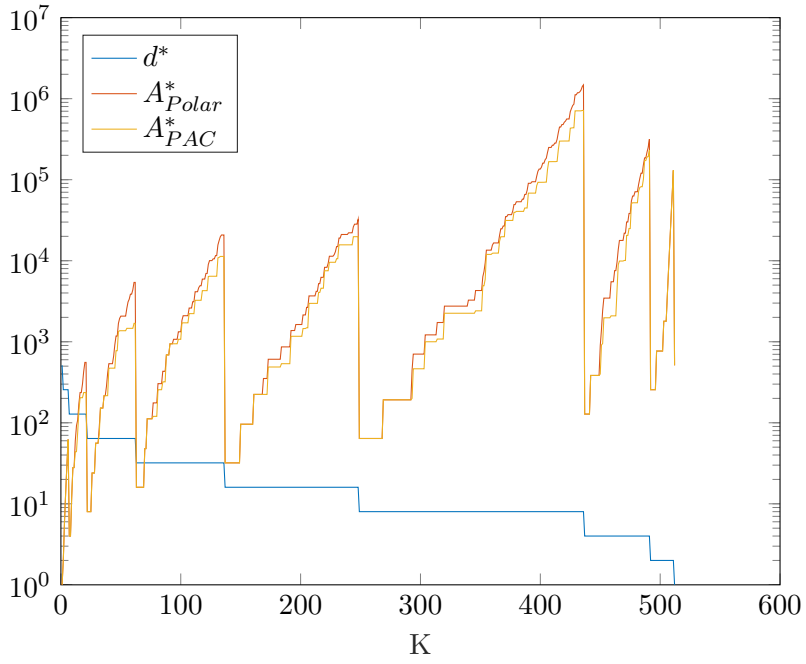


Figure 2.11: Representation of d^* et A^* for $N = 512$ and $\mathbf{g} = [1, 0, 1, 1, 0, 1, 1]$ depending on K

The computation of d^* and A^* for $N = 512$ and $K \in [\![1; N]\!]$ for the 5G standard frozen bit set is summarized in Figure 2.11 for both polar and PAC codes with a generator polynomial $\mathbf{g} = [1, 0, 1, 1, 0, 1, 1]$. The frozen bit sets are the ones specified in the 5G standard [9]. This proves that we are able to compute d^* and A^* for any desired code rate for a specific frozen bit set. Moreover, it can be observed that d^* is the same for polar and PAC codes for any code rate for a specific frozen set. The loss of performance as code rate increases is explained by the A^* value increase.
Results in Figure 2.11 also confirms that the number of occurrences of codewords with minimum weight for polar codes is greater or equal than for PAC codes. The cases where there is an equality in the number of occurrences of codewords with minimum weights result from the fact that all the codewords with minimum weights result from cosets $\mathcal{C}_N(\mathbf{0}_0^{i-1}, u_i = 1), i \in [\![s+1; N-1]\!]$. As for those cosets, the convolutional transformation has no effect, the number of occurrences remains the same.

N	128			256			512			1024		
R	1/4	1/2	3/4	1/4	1/2	3/4	1/4	1/2	3/4	1/4	1/2	3/4
s	113	97	66	226	197	145	465	417	385	961	897	770
$N - l_{CRC}$	117	117	117	245	245	245	501	501	501	1013	1013	1013

TABLE 2.6: Last enumeration step for pure and polar codes with CRC

2.3.4 Extension for polar codes with CRC

In this section, we leverage the previous findings for polar codes concatenated with a CRC. As the precoding of polar codes with a CRC differs from the one in the case of PAC codes or polar codes with DFB, we propose in this section a method to compute the minimum distance and its associated number of occurrence in the case of polar codes with CRC.

2.3.4.1 Representation of polar codes with CRC as union of polar cosets

Given a polar code with CRC, as the CRC bits are located at the end of the sequence. Therefore, a polar code concatenated with a CRC $\mathcal{C}_{CRC}$ can be expressed as:

$$\mathcal{C}_{CRC} = \bigcup_{\mathbf{u}_0^{N-l_{CRC}-1} \in L_{CRC}} \mathcal{C}_N^{(N-l_{CRC}-1)}(\mathbf{u}_0^{N-l_{CRC}-2}, u_{l_{CRC}-1}) \tag{2.45}$$

Where l_{CRC} denotes the length of the CRC and $\mathcal{L}_{CRC}$ denotes the set $\mathcal{L}_{CRC} = \left\{\mathbf{u}_0^{N-l_{CRC}-1} \in \{0,1\}^{N-l_{CRC}} | u_i = 0, \forall i \in \mathcal{F}\right\}$.
Table 2.6 represent the index of the last frozen bit s for pure polar codes and the value of $N - l_{CRC}$ for a 5G configuration using a CRC of length 11 from [9] for different block lengths and rates. It is observed that $N - l_{CRC}$ is higher than s and the difference is more significant for higher rates. This is one of the reasons why computing the minimum distance properties for polar codes with CRC is more complex than for pure polar codes. It is also important to note that as there is no direct way to determine the minimum distance for polar codes with CRC, the same incremental search presented in Algorithm 2 is maintained in order to determine the minimum distance.
Algorithm 3 gives details about the computation of minimum distance and related number of occurrences for polar codes with CRC. The algorithm operates as follows:

- The minimum distance d_{start} is initialized to the value of the minimum distance of the pure polar code.
- At each decoding step i, the different prefixes u_0^i are listed. w^* and A^* are computed

for each one of the cosets defined by the prefix u_0^i. The cosets with a minimum weight greater than the minimum distance are discarded.

- At the last enumeration step, i.e $i = N - l_{CRC}$, a CRC is generated for all the prefixes remaining in the list. The overall minimum distance of the remaining codewords $d^*(\mathcal{C}_{CRC})$ and number of occurrences $A^*(\mathcal{C}_{CRC})$ are computed.

- If $A^*(\mathcal{C}_{CRC}) = 0$, d_{start} is incremented and the whole process is repeated. When $A^*(\mathcal{C}_{CRC}) > 0$, the number of remaining codewords with minimum weight is obtained.

Algorithm 3: Computing d^* and A^* of a polar code with CRC

Input: Polar code with CRC $\mathcal{C}_{CRC}(N, K, \mathcal{F}, \mathbf{g}_{CRC})$

$L \leftarrow 1$
$\mathcal{L} \leftarrow \{0\}$ `/* List to store prefixes */`
$d_{start} \leftarrow d^*(\mathcal{C})$
$A^*(\mathcal{C}_{CRC}) \leftarrow 0$
while $A^*(\mathcal{C}_{CRC}) = 0$ **do**
  for $i \in [\![0; N - l_{CRC} - 1]\!]$ **do**
    for $l \in [\![1; L]\!]$ **do**
      Compute w^* of the cosets in the list
      Discard the cosets for which $w^* > d_{start}$
      $L \leftarrow |\mathcal{L}|$
    end
    if $i = N - l_{CRC} - 1$ **then**
      Generate the CRC for the remaining prefixes and compute the associated weights
      $A^* \leftarrow$ Occurrences of d_{start}
    end
  end
  Return $(d_{start}, A^*(\mathcal{C}_{CRC}))$
  $d_{start} \leftarrow d_{start} + 2$
end

Algorithm 3 consists in $N - l_{CRC}$ loop iterations where C_i cosets are evaluated at each loop. The complexity of the proposed method is driven by the total number of evaluated cosets n_r:

$$n_r = \sum_{i=0}^{N-l_{CRC}-1} C_i \tag{2.46}$$

N	l_{CRC}	R = 1/4			R = 1/2			R = 3/4		
		d^*	A^*	n_r	d^*	A^*	n_r	d^*	A^*	n_r
128	6	16	2	2717	8	19	9115	6	16	44164
	11	24	44	41359	12	6	254427	8	1794	108234166
	16	24	9	159906	12	5	1228279	6	38	10180408
256	6	32	914	621650	16	2728	4551547	8	2900	7719514
	11	32	69	1091711	16	191	11497563	8	178	26717927
	16	32	9	1483775	16	19	17482381	8	32	54780533
512	6	32	270	466565	16	2378	6954710	8	7919	8368192
	11	32	21	669854	16	81	7370891	8	87	8548390
	16	32	2	559627	16	12	18363311	8	12	12452226

TABLE 2.7: Minimum distance and number of occurrences parameters for polar codes with CRC

Table 2.7 summarizes the values of d^*, A^* and n_r for different block lengths, rates and CRC polynomials. The frozen bit sets and the CRC polynomials are the ones specified in the 5G standard. Compared to pure polar codes with the same rates, we can see from the table that the polar codes with CRC have larger minimum distances that pure polar codes. As an example, a $(128, 64)$ pure polar code has a minimum distance of 8, whereas the same polar code with CRC has a minimum distance of 12.
To better illustrate how the CRC enlarges the polar code minimum distance, we represent in Figure 2.12 the weight repartition of the cosets during the last 11 enumeration steps for a $(128, 64)$ polar code CRC11. For this specific code, the minimum distance is $d^* = 12$ and the associated number of occurrences is $A^* = 6$. This means that during the enumeration process, the cosets with weights lower or equal to 12 were kept in the list. At the last enumeration step (corresponding to 117), the CRC was applied on all the previously kept cosets. This implies that only one codeword is kept from every coset since the rest of codeword bits are determined by the CRC. When applying the CRC, the retained codewords are not necessarily those with the final minimum weight after the CRC check. Consequently, codewords with a weight of 8 are eliminated due to their failure to pass the CRC and the number of codewords with a weight of 12 is lowered.

In order to illustrate the evolution of the minimum distance and associated number of occurrences for polar codes with CRC, Figure 2.13 represents the evolution of d^*, A^* and C_{max} for a polar code with $N = 128$, a CRC11 from the 5G standard [9] depending on the rate.
Figure 2.13 shows that for each interval of codes having the same d^*, C_{max} is monotonic. However, it is noted that unlike for pure polar codes, PAC codes or polar codes with

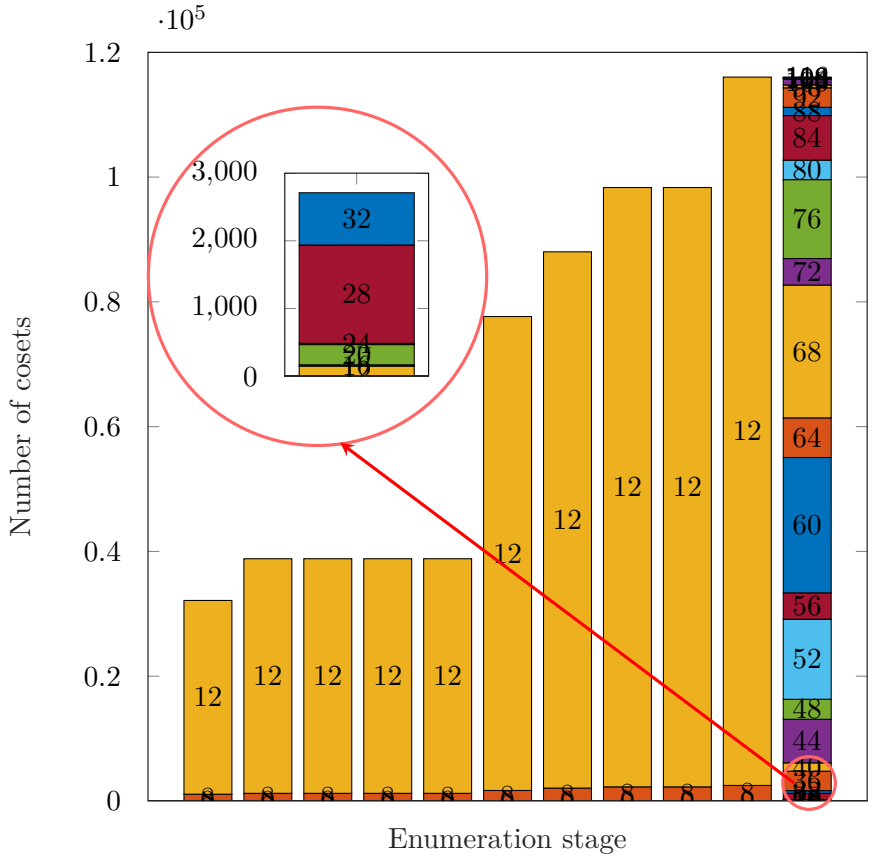


Figure 2.12: Weight distribution of the polar cosets of a $(128, 64)$ polar code with CRC11 during the last 11 enumeration steps

dynamic frozen bits, A^* is not necessarily monotonic for each interval of codes having the same d^*. This is due to the fact that the CRC bits are generated from $K - l_{CRC}$ bits and as K varies, the final codewords vary. In other terms, for pure polar codes, if $\mathbf{c}$ is a codeword of a (N, K) polar code, it is also a codeword of any $(N, K + K_i)$ such that $K_i \leq N - K$ polar code. But, it is not necessarily the case for polar codes with CRC. Table 2.8 contains the Average Visited Nodes (AVN) of Algorithm 3 and the PC-SCREM algorithm in [51] for $N = 128$ polar code constructed by PW and concatenated to a CRC of length 11 defined by 0xCBB. In our study case, the AVN of algorithm 3 is $n_r \times \log2(N)$. We can observe from the table that we are able to find the same values of d^* and A^*. It is also important to note that the number of AVN in our case is lower by several orders of magnitude.

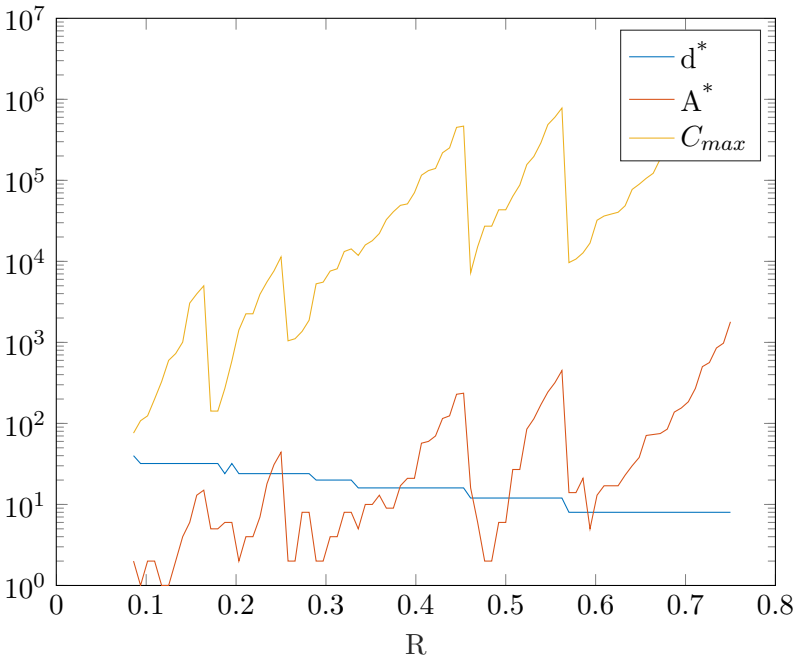


Figure 2.13: Evolution of d^*, A^* and C_{max} for a polar code with $N = 128$ and CRC11 depending on the code rate

N	R	d^*	A^*	AVN Algorithm 3	AVN [51]
128	1/4	16	3	37877	7×10^6
128	1/2	12	147	5749163	10^9
128	3/4	4	12	500983	3×10^6

TABLE 2.8: Minimum distance properties and AVN of $N = 128$ polar code with PW frozen set construction

2.4 Computation of the partial weight spectrum of pure and pre-transformed polar codes

As we showed in Section 2.2.2.3, it is possible to compute the RWEF of any polar coset for weights less or equal to w_{end}. This leads to a generalisation to Algorithms 1, 2 and 3 to enable the computation of the partial weight spectrum for pure and pre-transformed polar codes. Similarly to the MWEF of a polar code, the RWEF of a polar code $A_{\mathcal{C}}^{w_{end}}(X)$ can be expressed as:

$$A_{\mathcal{C}}^{w_{end}}(X) = \sum_{\mathbf{u}_0^s \in L_s} A_N^{w_{end}}(\mathcal{C}_N(\mathbf{u}_0^s))(X) \tag{2.47}$$

Algorithm 4 gives the details of the computation of the partial weight spectrum for polar codes. This algorithm can be generalized to pre-transformed polar codes as described in section 2.3.3. The main differences with the algorithm proposed to compute the minimum distance and the related number of occurrences are the following:

- Given a fixed w_{end}, at each enumeration step, the minimum weight of each coset is computed and the cosets with a minimum weight greater than w_{end} are discarded. At the first decoding step, it is not necessary to compute the whole spectrum for each coset since if the minimum distance of the coset is greater than the fixed threshold, then the coset contains no codewords with weights lower than the threshold and can therefore be discarded. As the computational complexity for computing the minimum distance of a coset is much lower than the computational complexity of computing the whole spectrum, this contributes in reducing the computational complexity further.

- At the last enumeration step, i.e when the last frozen bit is reached, the minimum weight and associated number of occurrences is computed for all the cosets remaining in the list. To achieve additional complexity reduction, the partial weight spectrum is calculated solely for cosets whose minimum weight is strictly lower than w_{end}.

Algorithm 4: Computing the partial spectrum of a pure polar code

Input: Polar code $\mathcal{C}(N, K, \mathcal{F}), w_{end}$

$s \leftarrow$ index of the last frozen bit
$L \leftarrow 1$
for $i \in [\![1; s]\!]$ **do**
    for $l \in [\![1; L]\!]$ **do**
        Compute w^* of the cosets in the list
        Discard the cosets for which $w^* > w_{end}$
        $L \leftarrow |\mathcal{L}|$
    end
end
if $i = s$ **then**
    Compute w^* of the cosets in the list
    Discard the cosets for which $w^* > w_{end}$
    Compute the RWEF of the remaining cosets in the list
end
Return $A_N^{w_{end}}(X)$

Figure 2.14 represents the overall algorithm's process for a $(32, 9)$ polar code with $w_{end} = 12$. The information set is defined by $\mathcal{I} = \{16, 23, 24, 26, 28, 29, 30, 31, 32\}$. We

can see from this figure that the overall RWEF is:

$$A_{\mathcal{C}}^{12}(X) = 1 + 20X^8 + 32X^{12} \tag{2.48}$$

The same approach described in sections 2.3.3 and 2.3.4 can be applied to adapt

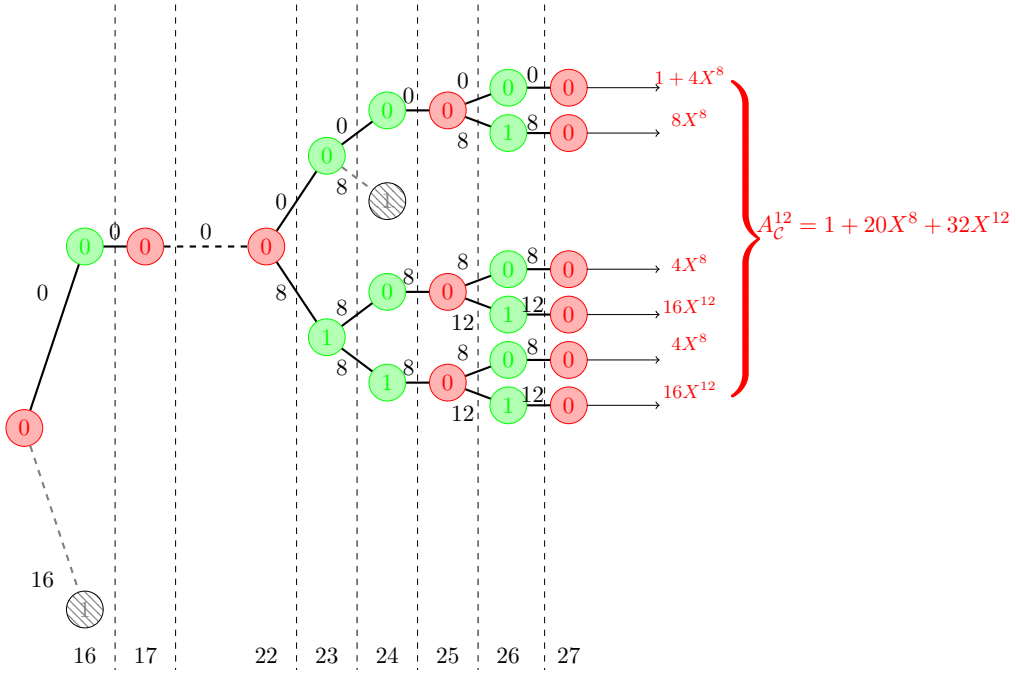


Figure 2.14: Relaxed spectrum enumeration for a $(32, 9)$ pure polar code

Algorithm 4 for computing the reduced spectrum of PAC codes, polar codes with DFB, or polar codes with CRC.

2.4.1 Partial distance spectrum results

This section presents an overview of the results obtained from computing the reduced polar spectrum for both pure and pre-transformed polar codes. Moreover, it includes a comparative analysis of the computational complexity associated with each spectrum generation technique presented in Algorithm 4 and the complexity of the algorithm proposed in [14]. In order to enable a fair comparison, we compare the computational complexity of Algorithm 4 to the time complexity of the algorithm proposed in [14].

Algorithm 4 consists in s loops in which the minimum weight of C_i cosets is evaluated at each enumeration step. At the last enumeration step, i.e. when $i = s$, the RWEF up to the weight w_{end} is computed for all the cosets remaining in the list except for the cosets whose minimum weight is equal to w_{end}. For those specific cosets, the RWEF

Code	(128,48)		(128,64)		(128,80)	
$\mathcal{S}_{w_{end}}$	w	A_w	w	A_w	w	A_w
	16	1817	8	288	8	4312
	20	17464	12	832	10	2368
	22	384	16	75864	12	374240
	24	311060	18	6272	14	1267648
	26	32128				
	28	3566024				
	30	1688167				
$TC^{w_{end}}$	45×10^9		1532×10^5		1.8×10^9	
$\Sigma\mathcal{N}^{(\mathcal{X})}_{Subcode}$ [14]	8672×10^9		3831×10^9		4585×10^9	

TABLE 2.9: Partial weight distribution of randomly pre-transformed polar codes

is equivalent to the MWEF. Let $C_s^{w_{end}}$ define the number of cosets having a minimum weight equal to w_{end} at the last enumeration step. The computational complexity $TC^{w_{end}}$ Algorithm 4 is at most equal to:

$$TC^{w_{end}} = \sum_{i=0}^{s} 6(N-1)C_i + 8{w_{end}}^2(N-1)(C_s - C_s^{w_{end}}) + 12(N-1)C_s^{w_{end}} \quad (2.49)$$

Table 2.10 provides an overview of the codeword distribution up to a specified weight for various Polar and PAC codes. Notably, the complexity of Algorithm 4 is significantly lower, often by several orders of magnitude, compared to the algorithm presented in [14] when the comparison is applicable. It's worth noting that the complexity value presented for Algorithm 4 represents the worst-case scenario and may be higher than the actual complexity. Additionally, the table demonstrates that PAC codes generally exhibit superior distance properties compared to polar codes. Table 2.9 presents the computed partial weight distribution for randomly pre-transformed polar codes, allowing for a comparison of the computational complexity of the partial weight spectrum computation for polar codes with DFB. It should be pointed out that as the precoding is performed randomly, exact reproducibility of values is not feasible, but values within a close range can be obtained. Comparing the overall complexity of Algorithm 4 with that of the algorithm in [14], we observe a significant reduction in computational complexity by several orders of magnitude. In [14], the overall running time for a C++ implementation on a computer with 6 cores i7 and a 3.2GHz processor is provided. In that case, the running time for a randomly pre-transformed $(128, 64)$ polar code is approximately one hour and a half. In contrast, our MATLAB implementation on a computer with 2 cores i5 and a 3.1GHz processor achieves a running time of less than 1 minute.

(N, K)	Type	(w, A_w)	$\Sigma\mathcal{N}^{(\mathcal{X})}_{Subcode}$ [14]	$TC^{w_{end}}$
$(128, 64)$	Polar	$(8, 304), (12, 768), (16, 161528)$ $(20, 4452096), (24, 166137744)$	-	6×10^9
	PAC	$(8, 256), (12, 960), (16, 76056)$ $(18, 18176), (20, 2455744), (22, 2857216)$ $(24, 86360192)$	-	24×10^9
$(256, 128)$	Polar	$(8, 96), (16, 111344), (20, 385024)$ $(22, 4452096), (24, 49907232)$	-	6×10^9
	PAC	$(8, 96), (16, 46320), (18, 3712)$ $(20, 430336), (22, 62976)$ $(24, 23598368)$	-	6×10^9
$(512, 256)$	Polar	$(16, 44640), (24, 12529152), (28, 95059968)$	20816×10^9	241×10^9
	PAC	$(16, 28000), (18, 384), (20, 16768)$ $(22, 3168), (24, 7706240), (26, 199360)$	-	120×10^9
$(1024, 512)$	Polar	$(16, 20672), (24, 2124800), (28, 262144)$	63×10^9	17×10^9
	PAC	$(16, 17472), (20, 768), (24, 1533696)$ $(28, 238)$	-	10×10^9
$(2048, 1024)$	Polar	$(16, 896), (32, 36067264)$	2195801×10^9	5799×10^9

TABLE 2.10: Partial weight distribution of pure and pre-transformed polar codes

2.5 Conclusion

In conclusion, this chapter presents a new low complexity algorithm designed to determine the minimum distance or more generally the partial weight spectrum of both pure and pre-transformed polar codes. In a first part, an approach that essentially relies on computing the distance properties of polar cosets is presented. Subsequently, each pure or pre-transformed polar code is expressed as a union of these disjoint cosets. By discarding the polar cosets that have no impact on the computation of the desired distance properties, this method ensures a deterministic computation, a crucial advantage over some of the existing techniques. The main advantages of the proposed method are the following:

1. This method operates independently of rate-profiling construction and imposes no restrictions on it, in contrast to other methods where the information set has to obey to the partial order constraint. This is because the computation of the MWEF and RWEF of polar cosets does not depend on any particular structure.
2. This method can be applied to pure polar codes but also extented to PAC codes, polar codes with dynamic frozen bits or polar codes concatenated with a CRC. In all of the mentioned cases, the presented method fully adapts to the choice of the precoding.
3. Comparative analysis with established methods underscores the superiority of this approach, demonstrating lower computational complexity in comparison to existing methods in the literature

We present in Chapter 3 a generalization if this work enabling the computation of the reduced spectrum of pure, pre-transformed punctured and shortened polar codes. The objective is to propose in the same optic, a method that can adapt to every puncturing and/or shortening pattern.

3 About the distance properties of Punctured and Shortened pure and pre-transformed polar codes

This chapter focuses on the determination of the distance properties of pure and pre-transformed punctured and shortened polar codes. Our contributions consist in two parts: 1) Computation of the distance properties of what we define as rate-compatible polar cosets, 2) Introducing a low-complexity algorithm that computes the overall desired distance properties of pure and pre-transformed punctured and shortened polar codes.

3.1 Context

The Ultra-Reliable Low Latency Communication (URLLC) scenario poses stringent requirements for both ultra-low latency and high error correction performance approaching maximum likelihood (ML) decoding. Achieving these objectives is particularly challenging for short blocklengths. But it is essential to meet the expectations in terms of ultra-low latency without significant impact of performance degradation.

Usually, polar codes are constructed with sizes that are powers of 2. It limits their applicability in scenarios requiring short blocklengths. To overcome this limitation, various techniques such as puncturing and shortening have been introduced [66–71]. These methods enable the construction of rate-compatible polar codes. with more flexible code lengths.

The determination of the distance properties of punctured and shortened polar codes is crucial for optimizing their performance under maximum likelihood decoding. This knowledge allows for the optimization of puncturing and shortening patterns, as well as the associated rate-profiling, to achieve desired performance objectives. Additionally, considering the impact of pre-transformation on punctured and shortened polar codes is essential for enhancing their distance properties and ultimately building more reliable codes. Investigating how pre-transformation techniques affect the distance properties of punctured and shortened codes provides insights into how to design more robust and efficient coding schemes. Authors of [14] introduce a method for determining the partial weight spectrum of shortened and punctured polar codes. While this method offers some innovative insights, its practical utility is hindered by its significant computational complexity. Furthermore, extending the method to pre-transformed punctured and shortened polar codes exacerbates this issue, particularly because the original approach already suffers from high computational complexity when applied to pre-transformed polar codes. This chapter is dedicated to the computation of the distance properties of punctured and shortened pure and pre-transformed polar codes.

3.2 Rate-compatible pure and pre-transformed polar codes

3.2.1 Punctured polar codes

A punctured polar code denoted by $\mathcal{C}_P(N_p, K, \mathcal{F})$ is obtained from a parent polar code $\mathcal{C}(N, K, \mathcal{F})$ by not transmitting $N - N_p$ codeword bits. We denote by $\mathcal{P}$ the puncturing pattern and $P = |\mathcal{P}| = N - N_p$. $\mathcal{P}$ describes the indexes of the non-transmitted codeword bits, i.e., the transmitted codeword is $\mathbf{x}_{\bar{\mathcal{P}}}$.

From the decoder's perspective, the punctured codeword bits are considered as erased. Therefore, they are totally unknown to the decoder. The LLRs associated to the punctured positions are therefore set to zero. It is important to note that the unreliability of the codeword bits $\mathbf{x}_{\mathcal{P}}$, caused by the fact of being unknown to the decoder, is transmitted to the initially transmitted vector $\mathbf{u}$. Thus, some of the bits of $\mathbf{u}$ are considered as *incapable* [69], i.e. bits that have an LLR equal to 0. Therefore, it is not possible to take a decision for those specific bits. We define the set of incapable bits as follows:

Definition 3.2.1. *Let $\mathcal{C}_P(N_p, K, \mathcal{F})$ be a punctured polar code by the pattern $\mathcal{P}$. We denote by $\mathcal{U}_{\mathcal{P}}$ the set of incapable bits such as:*

$$i \in \mathcal{U}_{\mathcal{P}} \iff L_i = 0 \tag{3.1}$$

It has been showed in [72] that:

$$|\mathcal{U}_{\mathcal{P}}| = |\mathcal{P}| = P \tag{3.2}$$

In other words, puncturing P codeword bits generates exactly P incapable bits. It is therefore essential to freeze the incapable bits as their LLR values cannot be resolved during SC decoding. Indeed, this can lead to an error floor from the performance point of view since their associated LLRs are equal to 0 and a decision cannot be made. In all the following, we will denote by $\mathcal{F}_{\mathcal{P}} = \mathcal{F} \bigcup \mathcal{U}_{\mathcal{P}}$ the set of frozen bits for punctured polar codes.
It is important to point out that any method proposed for computing the frozen set for punctured polar codes [66–69] guarantees that $\mathcal{U}_{\mathcal{P}} \subseteq \mathcal{F}_{\mathcal{P}}$. One of the major open questions regarding punctured polar codes is the choice of the puncturing pattern $\mathcal{P}$ that guarantees better performance for punctured polar codes. It is essential to note that in the case of punctured polar codes, the choice of the information set heavily depends on the choice of the puncturing pattern $\mathcal{P}$.

In this context, various puncturing techniques have been introduced to address this challenge. One approach involves using density evolution to determine $\mathcal{F}_{\mathcal{P}}$ [66, 67]. In this case, a given puncturing pattern is used to simulate channel polarization through density evolution, enabling the identification of the incapable bits. Subsequently, the code is constructed by selecting the optimal information set adapted to the puncturing scheme. It's worth noting that this method requires to optimize the information set based on density evolution for each channel output. Alternatively, simpler puncturing patterns have been proposed. For instance, in [68], a Quasi-Uniform Puncturing (QUP) method was introduced. Additionally, [69] presented a low-complexity puncturing technique for

polar codes based on the Bit-Reversal permutation (BRP).

3.2.2 Shortened polar codes

As for punctured polar codes, a shortened polar code denoted by $\mathcal{C}_S(N_s, K, \mathcal{F})$ is obtained from a parent polar code $\mathcal{C}(N, K, \mathcal{F})$ by removing $N - N_s$ codeword bits. The difference resides in the fact that unlike for punctured polar codes, the non-transmitted codeword bits in the case of shortened polar codes are set to a known value, usually 0. In all the following, we will fix that shortened codeword bits have to be set to zero. We denote by $\mathcal{S}$ the shortening pattern and $S = |\mathcal{S}| = N - N_s$. $\mathcal{S}$ describes the indexes of the cancelled codeword bits. This means that the transmitted codeword is $\mathbf{x}_{\bar{\mathcal{S}}}$ and $\mathbf{x}_{\mathcal{S}} = \mathbf{0}$.

From the decoder's perspective, the shortened codeword bits are perfectly known to be equal to zero. Therefore, their associated LLR values are set to infinity or in practical to a large positive value. Since the shortened codeword bits are perfectly known to the decoder, they lead to very reliable elements of $\mathbf{u}$ qualified as *overcapable* [69]. We define the set of overcapable bits as:

Definition 3.2.2. *Let $\mathcal{C}_S(N_s, K, \mathcal{F})$ be a shortened polar code by the pattern $\mathcal{S}$. We denote by $\mathcal{U}_{\mathcal{S}}$ the set of overcapable bits such as:*

$$i \in \mathcal{U}_{\mathcal{S}} \iff L_i = \infty \tag{3.3}$$

In practice, the condition $\mathbf{x}_{\mathcal{S}} = \mathbf{0}$ is only satisfied if $u_i = 0$, i.e. frozen $\forall i \in \mathcal{U}_{\mathcal{S}}$. We denote by $\mathcal{F}_{\mathcal{S}} = \mathcal{F} \bigcup \mathcal{U}_{\mathcal{S}}$ the set of frozen bits in the case of shortened polar codes.

In the case of systematic polar codes [73], shortening is straightforward. However, for non-systematic polar codes, shortening is more complex since it involves determining the exact S bits to set to zero (or frozen) to ensure $\mathbf{x}_{\mathcal{S}} = \mathbf{0}$. Shortening was initially introduced as a puncturing technique in [74]. The proposed method starts with the matrix $\mathbf{G}_N$, identifies columns with weight 1, selects a column that meets this criterion, adds its index to the shortening pattern, and finally removes the row and column corresponding to the position of "1". This process is iterated S times. In [74], shortening of the final bits of the mother polar code was suggested, as it follows the aforementioned procedure. Alternatively, in [75], Dynamic Frozen Bits were used to ensure $\mathbf{x}_{\mathcal{S}} = \mathbf{0}$. A joint optimization of the shortening set and the remaining frozen set were introduced in this article to identify the most suitable rate-profiling for a shortened polar code. Additionally, a shortening pattern based on the Bit-Reversal Permutation is proposed in [69] .

3.2.3 Rate-Compatible Pre-Transformed polar codes

In the context of pre-transformed polar codes, the mapping $\mathbf{u} = \mathbf{vT}$ where $\mathbf{T}$ is an upper triangular matrix is considered. Finally, authors of [71] introduced symmetric puncturing strategies that have been proved to outperform the aforementioned puncturing techniques.

In the case of punctured polar codes, as the values of punctured codeword bits are considered erased and unknown, the precoding step does not affect puncturing. Therefore, any puncturing pattern that is proposed for polar codes can be used similarly for pre-transformed polar codes.

However, in the case of shortened polar codes , the shortening constraint implies that $\mathbf{x}_{\mathcal{S}} = \mathbf{0}$. This constraint cannot be guaranteed in the case of shortened pre-transformed polar codes, as the fact of ensuring that $\mathbf{v}_{\mathcal{U}_{\mathcal{S}}} = \mathbf{0}$ does not necessarily lead to $\mathbf{u}_{\mathcal{U}_{\mathcal{S}}} = \mathbf{0}$.

In this context, a constraint is proposed in [76] for the pre-transformation for shortened polar codes. The constrained pre-transformation consists in:

$$u_i = \begin{cases} v_i & \text{if} \quad i \in \mathcal{U}_{\mathcal{S}} \\ \sum_{j=0}^{m-1} g_j v_{i-j} & \text{otherwise} \end{cases} \tag{3.4}$$

By enforcing $\mathbf{v}_{\mathcal{U}_{\mathcal{S}}} = \mathbf{u}_{\mathcal{U}_{\mathcal{S}}} = \mathbf{0}$, the condition $\mathbf{x}_{\mathcal{S}}$ is achieved. In all the following, we will consider the constraint given in equation (3.4) for the shortened pre-transformed polar codes.

One of the main open questions regarding rate-compatible pure and pre-transformed polar codes is the rate-profile construction. For this reason, in this chapter, we propose a low complexity algorithm capable of computing the minimum distance and the associated number of occurrences of rate-compatible pure and pre-transformed polar codes. This approach is an extension to the one detailed in Chapter 2. It offers the advantage of not assuming any specific structure (a) for the frozen bit set, (b) for the pre-transformation or (c) for the shortening and puncturing patterns. It, therefore, aids in the design of puncturing shortening patterns, pre-transformation parameters, and frozen bit sets for punctured/shortened polar codes, thereby enhancing their decoding performance.

3.3 Computing the minimum weight of rate-compatible polar cosets

In this section, the concept of rate-compatible Polar Cosets is introduced. We can adapt an approach introduced in [77], which enables the computation of the minimum weight of polar cosets, to calculate the minimum weight of rate-compatible cosets. We detailed in Chapter 2 the principle of message-passing rules applied to compute the minimum weight of a given polar coset. In this section, we modify the way of defining polar cosets in the case of rate-compatible polar codes in order to take the puncturing or/and the shortening effect into account. We define rate-compatible punctured and shortened polar cosets respectively as:

$$\begin{cases} \mathcal{CP}_N(\mathbf{u}_0^i) = \{\mathbf{x}_{\bar{\mathcal{P}}} | \mathbf{x} \in \mathcal{C}_N(\mathbf{u}_0^i)\} \\ \mathcal{CS}_N(\mathbf{u}_0^i) = \{\mathbf{x}_{\bar{\mathcal{S}}} | \mathbf{x} \in \mathcal{C}_N(\mathbf{u}_0^i), \mathbf{x}_{\mathcal{S}} = \mathbf{0}\} \end{cases} \tag{3.5}$$

where $\bar{\mathcal{V}}$ denotes the complement of the set $\mathcal{V}$. This representation is different from $\mathcal{C}_N^{(i)}(\mathbf{u}_0^i)$ in the way that it takes the effect of puncturing or shortening into account. It is thus possible to compute $w^*(\mathcal{CP}_N(\mathbf{u}_0^i))$ and $w^*(\mathcal{CS}_N(\mathbf{u}_0^i))$ using an approach that is similar to the one used to define the LLR values of rate-compatible polar codes.

In the case of punctured polar codes, given x_i such that $i \in \mathcal{P}$, the value of x_i is erased. Therefore, it does not play any role into the determination of the different codewords weights. When taking this into consideration, each leaf node $x_i, i \in \mathcal{P}$ is initialised as follows:

$$\boldsymbol{\mu}_{x_i} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{3.6}$$

This is explained by the fact that the value of x_i is not taken into consideration whether it is equal to 0 or 1 since it is not transmitted.

Example 3.3.1. *An illustration of a distance factor graph of bit u_4 is given by Figure 3.1. We consider configurations for a punctured polar code. In this configuration, $N = 8$, $K = 4$ and $P = 2$. The puncturing pattern is $\mathcal{P} = \{0, 4\}$ and the frozen bit set is $\mathcal{F} = \{0, 1, 2, 4\}$. The evaluated cosets are $\mathcal{CP}_8([0, 1, 0, 0], 0)$ and $\mathcal{CP}_8([0, 1, 0, 0], 1)$, thus $\mathbf{p} = [0, 1, 0, 0]G_0^3 = [1, 1, 0, 0, 0, 0, 0, 0]$. In the case of the punctured codeword bits x_0 and x_4, they are initialised to $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$, the rest are initialized according to (2.15). The message passing is based on Equations (2.11) and (2.14) We can see from Figure 3.1 that*

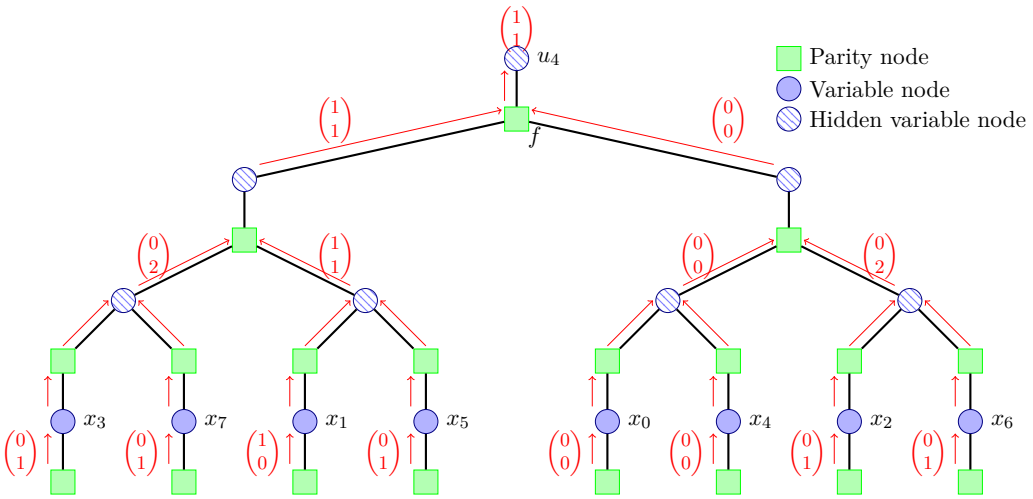


Figure 3.1: Tanner Graph of u_4 decoding for a punctured polar code with $N = 8$ and $P = 2$

$$\boldsymbol{\mu}_{f\to u_4} = \begin{pmatrix} w^*(\mathcal{C}_8([0,1,0,0],0)) \\ w^*(\mathcal{C}_8([0,1,0,0],1)) \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

In the case of shortened polar codes, the shortening pattern is defined to guarantee that $x_i = 0, i \in \mathcal{S}$. This means that there are no codeword configurations with $x_i = 1$. This leads to the following initialization for every leaf node x_i such that $i \in \mathcal{S}$:

$$\boldsymbol{\mu}_{x_i} = \begin{pmatrix} 0 \\ \infty \end{pmatrix} \tag{3.7}$$

As the second row of $\boldsymbol{\mu}_{x_i}$ defines the weight of the relative configuration to x_i when x_i is equal to 1, setting it to infinity (or in practical to a large positive value) prevents it from appearing in any configuration of minimum weight.

Example 3.3.2. *An illustration of a distance factor graph of bit u_3 is given by Figure 3.2. We consider configurations for a shortened polar code. In this configuration, $N = 8$, $K = 4$ and $S = 2$. The shortening pattern is $\mathcal{S} = \{3, 7\}$ and the frozen bit set is $\mathcal{F} = \{0, 1, 3, 7\}$. The evaluated coset are $\mathcal{CS}_8([0,1,0], 0)$ and $\mathcal{CS}_8([0,1,0], 1)$, thus $\mathbf{p} = [0,1,0]G_0^2 = [1,1,0,0,0,0,0,0]$. In the case of the shortened codeword bits x_3 and x_7, they are initialised to $\begin{pmatrix} 0 \\ \infty \end{pmatrix}$, the rest are initialized according to (2.15). The message passing is based on Equations (2.11) and (2.14) We can see from Figure 3.2. that $\boldsymbol{\mu}_{u_3} = \begin{pmatrix} 2 \\ \infty \end{pmatrix}$. This means that $w^*(\mathcal{CS}_8([0,1,0], 0)) = 2$ for $u_3 = 0$. On the other*

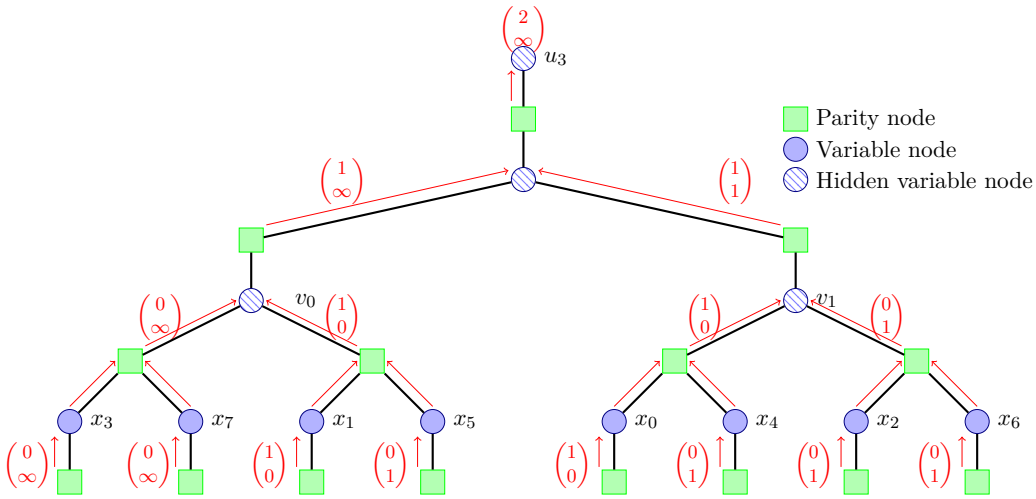


Figure 3.2: Tanner Graph of u_3 decoding for a shortened polar code with $N = 8$ and $S = 2$

hand, $w^*(\mathcal{CS}_8([0,1,0],1)) = \infty$*. This reflects the fact that the initialization takes into account the absence of configurations where* $u_3 = 1$ *in the shortened polar code, as* u_3 *is constrained to be 0 by the shortening pattern.*

3.4 Weight enumeration function of rate-compatible polar cosets

We showed in Section 2.2.2.2 that it is possible to simultaneously compute the minimum weight and it related number of occurrences for a polar coset. We show in this section that similar rules can be applied in order to compute the minimum weight and associated number of occurrences for a rate-compatible polar coset.

3.4.1 Case of punctured polar codes

Let $\mathcal{CP}_N(\mathbf{u}_0^i)$ denote a punctured polar coset and by $A_N^*(\mathcal{CP}_N(\mathbf{u}_0^i))(X)$ the MWEF of $\mathcal{CP}_N(\mathbf{u}_0^i)(X)$. The aim is to compute $A_N^*(\mathcal{CP}_N(\mathbf{u}_0^i))(X)$. In this case, the initial messages that are sent from the leaf nodes representing the MWEF of each x_i is:

$$\boldsymbol{\theta}_{x_i} = \begin{pmatrix} 1X^0 \\ 1X^0 \end{pmatrix} \tag{3.8}$$

This is explained by the fact that both configurations of x_i for which $x_i = 0$ and $x_i = 1$ do not affect the final codeword weight since it is erased. As there is no need to change the graph structure, the message passing equations (2.24) and (2.26) can be applied to compute the MWEF of a punctured polar codes. However, there is a modification that needs to be taken into consideration in the case of punctured polar codes. Actually, as a punctured polar coset $\mathcal{CP}_N(\mathbf{u}_0^i))$ describes the affine space generated by the punctured last $N-i-1$ rows of the generator matrix, the resulting punctured matrix may not be full rank due to puncturing. Therefore, in the case of punctured polar codes, the final MWEF of a punctured polar coset has to be divided by $\frac{1}{2^{N-i-1-rk(GP_{i+1}^{N-1})}}$. where $rk(.)$ computes the rank of a matrix. We can see that when the matrix GP_{i+1}^{N-1} is full rank, then $\frac{1}{2^{N-i-1-rk(GP_{i+1}^{N-1})}} = 1$.

Example 3.4.1. *An illustration of a distance factor graph of bit u_3 is provided by Figure 3.3. Configurations for a punctured polar code are considered. In this configuration, we consider the parameters $N = 8$, $K = 2$ and $P = 4$. The puncturing pattern $\mathcal{P} = \{0, 2, 4, 6\}$ and the frozen bit set $\mathcal{F} = \{0, 1, 2, 3, 4, 6\}$. The evaluated cosets are and $\mathcal{CP}_8([0, 0, 0], 1)$, thus $\mathbf{p} = [0, 0, 0]G_0^2 = [0, 0, 0, 0, 0, 0, 0, 0]$.*

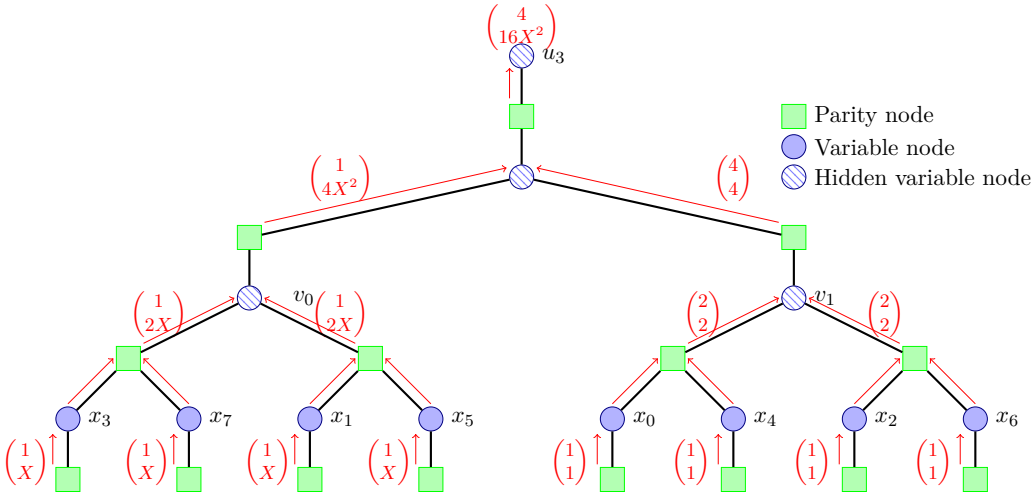


Figure 3.3: Tanner Graph of u_3 decoding for a punctured polar code with $N = 8$ and $P = 2$

In this case, the cosets $\mathcal{CP}_8([0, 0, 0], 0)$ and $\mathcal{CP}_8([0, 0, 0], 1)$ describe the space generated by the matrix GP_4^7 given in Figure 3.4. We can see from Figure 3.4 that due to the presence of two zero rows, the rank of the matrix is equal to 2 instead of 4 when taking the puncturing into account. This means that $A_N^*(\mathcal{CP}_8([0, 0, 0], 0))(X) = \frac{1}{2^{8-3-1-2}}.4 = 1$ and $A_N^*(\mathcal{CP}_8([0, 0, 0], 1))(X) = \frac{1}{2^{8-3-1-2}}.16X^2 = 4X^2$. The vectors in $\mathcal{CP}_8([0, 0, 0], 1)$ are enumerated in Figure 3.4. This corroborates the obtained result, i.e. there are four

$$\begin{matrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{matrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Figure 3.4: Transformation matrix for $N = 8$ and $P = 4$

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Figure 3.5: Vectors of $\mathcal{CP}_8([0,0,0],1)$ with minimum weight

distinct vectors whose minimum weight is equal to 2.

3.4.2 Case of shortened polar codes

Let $\mathcal{CS}_N(\mathbf{u}_0^i)$ denote a shortened polar coset. The aim is to compute $A_N^*(\mathcal{CS}_N(\mathbf{u}_0^i))(X)$. In this case, the initial messages that are sent from the leaf nodes representing the MWEF of each x_i is:

$$\boldsymbol{\theta}_{x_i} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{3.9}$$

This is explained by the fact that there is no configuration for which a shortened codeword bit x_i is equal to 1 since this is imposed by the shortening pattern. The term 0 can be interpreted as "$0X^\infty$".
The same message passing equations (2.24) and (2.26) are applied to compute the MWEF

of shortened polar cosets. In the case of shortened polar codes and due to the shortening constraints described in [74], the matrix GS_{i+1}^{N-1}, has a full rank. Therefore, unlike puncturing, no further operations are required in order to compute the MWEF.

Example 3.4.2. *An illustration of a distance factor graph of bit u_3 is provided by Figure 3.6. Configurations for a shortened polar code are considered. In this configuration, we consider the parameters $N = 8$, $K = 4$ and $S = 2$. The shortening pattern is $\mathcal{S} = \{3, 7\}$ and the frozen bit set is $\mathcal{F} = \{0, 1, 3, 7\}$. The evaluated cosets are $\mathcal{CS}_8([1, 0, 0], 0)$ and $\mathcal{CS}_8([1, 0, 0], 1)$, thus $\mathbf{p} = [1, 0, 0]G_0^2 = [1, 0, 0, 0, 0, 0, 0, 0]$. In the case of the shortened*

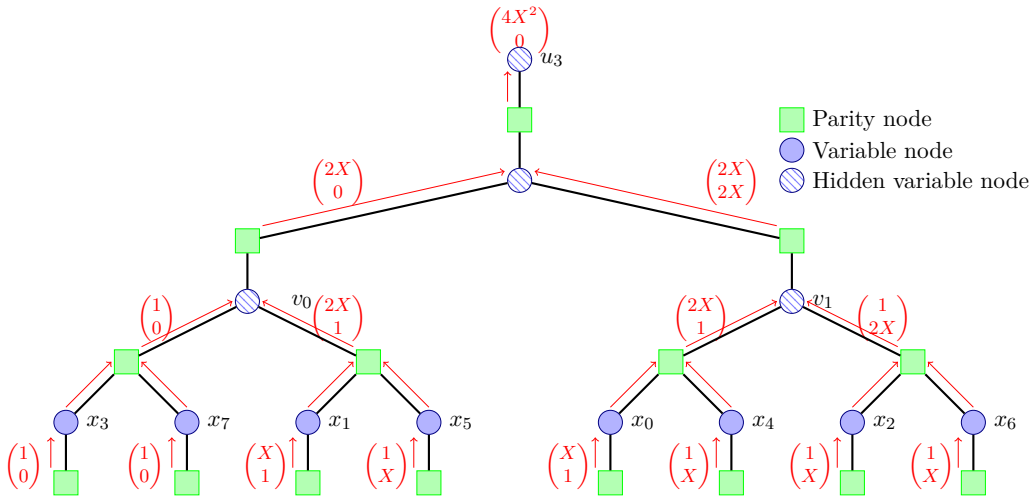


Figure 3.6: Tanner Graph of u_3 decoding for a shortened polar code with $N = 8$ and $S = 2$

codeword bits x_3 and x_7, they are initialised to $\begin{pmatrix}1\\0\end{pmatrix}$, the rest are initialized according to (2.27). The message passing is based on Equations (2.24) and (2.26) We can see from Figure 3.6. that $\boldsymbol{\theta}_{u_3} = \begin{pmatrix}4X^2\\0\end{pmatrix}$. This means that $A_N^*(\mathcal{CS}_N([0, 1, 0], 0))(X) = 4X^2$ for $u_3 = 0$. Consequently, there are 4 codewords of minimum weight 2. On the other hand, $A_N^*(\mathcal{CS}_N([0, 1, 0], 1))(X) = 0$. This underscores that the initialization considers the absence of configurations in the shortened polar code where $u_3 = 1$, as u_3 is forced to be equal 0 by the shortening pattern.

3.5 Extension to the Reduced Weight Enumeration Spectrum of punctured and shortened polar cosets

Similarly to Section 2.2.2.3, it is possible to extend the computation of the MWEF of a rate-compatible polar coset to the computation of the RWEF up to a fixed weight w_{end}

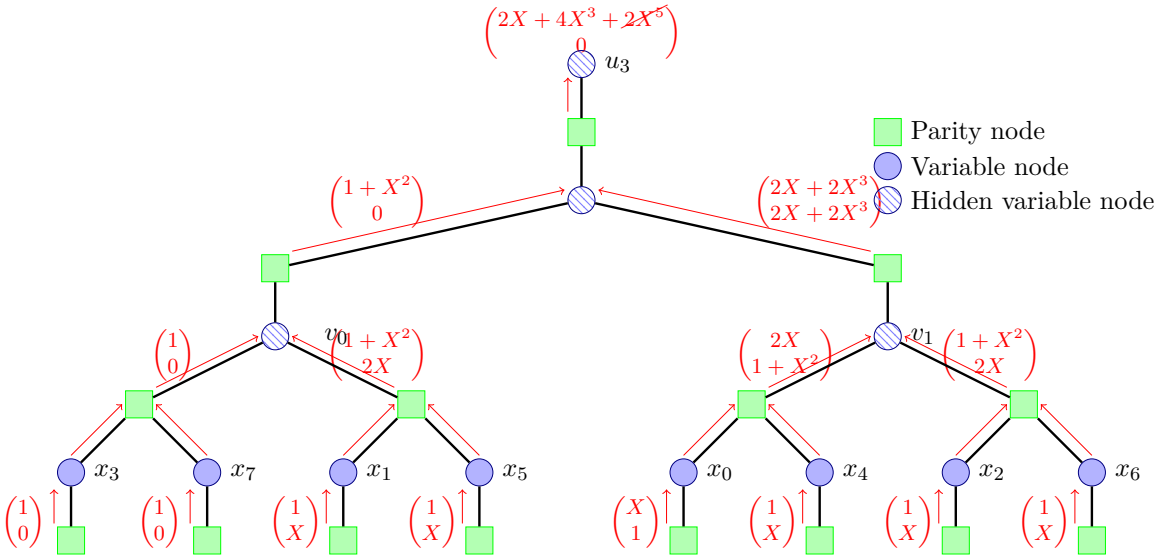


Figure 3.7: Factor graph of u_3 for the computation of $A_8^3(\mathcal{CP}_8([1,0,0],0))$ and $A_8^3(\mathcal{CP}_8([1,0,0],1))$

thanks to the same relations in (2.31) and (2.33). We denote by $A_N^{w_{end}}(\mathcal{CP}_N(\mathbf{u}_0^i))(X)$ and $A_N^{w_{end}}(\mathcal{CS}_N((\mathbf{u}_0^i))(X)$ the RWEF for weight lower or equal to w_{end} of a punctured and shortened polar coset respectively.

In the case of punctured polar cosets, where puncturing alters the generator matrix and may lead to non-full rank configurations, adjustments are necessary to adapt the RWEF for accurate weight spectrum computation similarly to the MWEF. This implies that the coefficients of the RWEF of a punctured polar coset $A_N^{w_{end}}(\mathcal{CP}_N((\mathbf{u}_0^i))$ has to be divided by $\frac{1}{2^{N-i-1-rk(GP_{i+1}^{N-1})}}$.
In the case of a shortened polar coset, no further adaptation need to be undertaken.

Example 3.5.1. *An illustration of the computation of the RWEFs* $A_8^3(\mathcal{CP}_8([1,0,0],0))$ *and* $A_8^3(\mathcal{CP}_8([1,0,0],1))$ *for a polar code of length* $N = 8$ *is provided by Figure 3.7. In this case, we consider* $\mathbf{p} = [1,0,0,0,0,0,0,0]$ *and* $w_{end} = 3$*. The computation is performed by using Equation* (2.31) *in the case of a parity node and Equation* (2.33) *in the case of a check node. We can see from the figure that at each computation step, the monomials with a degree greater than* 3 *is eliminated.*

$$
\begin{array}{c}
u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \\ u_{10} \\ u_{11} \\ u_{12} \\ u_{13} \\ u_{14} \\ u_{15}
\end{array}
\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{pmatrix}
$$

Figure 3.8: Transformation matrix $\mathbf{G}_{16}$.

3.6 Computing the distance properties of Rate-Compatible pure and pre-transformed polar codes

In this section, we take advantage of the results on the computations of the minimum weight as well as the MWEF of rate-compatible polar cosets to compute the overall minimum distance and number of codewords of minimum distance of rate-compatible pure and pre-transformed polar codes.
Similarly to equation (2.34), starting with an example, we show that in the case of a rate-compatible polar code, $\mathcal{CV}$ can also be written as the disjoint union of the following cosets:

$$\mathcal{CV} = \bigcup_{\mathbf{u}_0^{s-1} \in L_s} \mathcal{CV}_N^{(s)}(\mathbf{u}_0^{s-1}, u_s = 0) \tag{3.10}$$

Where $s = \max(\mathcal{F})$. Note that s is the last frozen bit of $\mathcal{F}$ not of $\mathcal{F}_V$, i.e., it only considers the bits that has not been frozen due to the puncturing/shortening constraint.

Example 3.6.1. *Let us consider the polar code shortened from the parent code (16,7) using the bit-reversal permutation shortening pattern of length 2. The transformation matrix* $\mathbf{G}_{16}$ *is shown in Figure 3.8. The rows and columns deleted of the transformation matrix due to the shortening are highlighted in green.*

In this figure, the frozen bits of indexes $\mathcal{F} = \{0, 1, 2, 3, 4, 5, 8\}$*, are represented in black,*

the shortened bits of indexes $\mathcal{S} = \{7, 15\}$ in green. The remaining information bits are represented in red for the ones before the last frozen bit and in blue for the ones after the last frozen bit.
Let us consider the shortened polar coset $\mathcal{CS}_{16}(\mathbf{u}_0^8)$. For any $\mathbf{u}_0^8$ such that $u_i = 0 \quad \forall i \in \mathcal{F}$, and $u_i = \{0, 1\} \quad \forall i \in \mathcal{I}$, the shortened polar coset $\mathcal{CS}_{16}(\mathbf{u}_0^8)$ forms a subset of the shortened polar code $\mathcal{CS}$. In fact, by definition of $\mathcal{CS}_{16}(\mathbf{u}_0^8)$, the shortened bits u_7 and u_{15} are constrained to be zero. Therefore, the total shortened polar code $\mathcal{C}_\mathcal{S}$ can be described as:

$$\mathcal{CS} = \bigcup_{u_0^8 \in L_8} \mathcal{CS}_{16}(\mathbf{u}_0^8) \tag{3.11}$$

where $L_8 = \left\{ \mathbf{u}_0^7 \in \{0,1\}^8 \,|u_i = 0 \quad \forall i \in \mathcal{F} \right\}$.

Based on Equation (3.10), we can apply the same algorithm described in Chapter 2 in order to compute the number of codewords with minimum distance or more generally the reduced weight spectrum for rate-compatible pure and pre-transformed polar codes. Note that Equation (3.10) can also be generalised to pre-transformed polar codes as it was shown in Chapter 2. In this context, the MWEF of a punctured or shortened polar code $A^*_{\mathcal{C}\setminus\{\mathbf{0}\}}(X)$ can be expressed as:

$$A^*_{\mathcal{CV}\setminus\{\mathbf{0}\}}(X) = \mathrm{LP}\left(\sum_{\substack{\mathbf{u}_0^s \in L_s \\ \mathbf{u}_0^s \neq \mathbf{0}_0^s}} A^*_N(\mathcal{CV}_N(\mathbf{u}_0^s))(X) + A^{d^*}_N(\mathcal{CV}_N(\mathbf{0}_0^s)(X)) - 1 \right) \tag{3.12}$$

Similarly, the RWEF up to a weight w_{end} of a punctured or shortened polar code $A^{w_{end}}_\mathcal{C}(X)$ is expressed as:

$$A^{w_{end}}_{\mathcal{CV}}(X) = \sum_{\mathbf{u}_0^s \in L_s} A^{w_{end}}_N(\mathcal{CV}_N(\mathbf{u}_0^s))(X) \tag{3.13}$$

Algorithms 1 and 2 summarise the computation of the reduced weight spectrum in the case of punctured and shortened PAC codes respectively. The algorithms operate as follows:

- For each $i \in [\![0, s]\!]$, all the prefixes $\mathbf{u}_0^i = \mathbf{g}(\mathbf{v}_0^i)$ that remained in the list at an exploration stage i are listed.

- For each of the aforementioned prefixes, $w^*(\mathcal{CV}_N(\mathbf{u}_0^i))$ is computed.
- The cosets with $w^*(\mathcal{CV}_N(\mathbf{u}_0^i)) > w_{end}$ are discarded. Those cosets are irrelevant to the computation the partial weight spectrum with the threshold on weight w_{end} as:

$$w^*(\mathcal{CV}_N(\mathbf{u}_0^i, u_{i+1})) \geqslant w^*(\mathcal{CV}_N(\mathbf{u}_0^{i-1}, u_i)) \tag{3.14}$$

  This means that $\forall \quad \mathcal{CV}_N(\mathbf{u}_0^i, u_{i+1})$ with $w^*(\mathcal{CV}_N(\mathbf{u}_0^i, u_{i+1})) < w_{end}$, $\nexists j \in [\![i+1; N-1]\!]$ such that $w^*(\mathcal{CV}_N(\mathbf{u}_0^{j-1}, u_j)) = w_{end}$. In other words, if a coset has a minimum weight $w > w_{end}$, then no codeword within that coset can have a weight lower than or equal to w_{end}.

- When $i = s$, the partial weight spectrum is obtained by computing the RWEF of the cosets that remained in the list. The sum of the RWEFs results in the RWEF of the punctured/shortened PAC code as shown in Equation (3.13).

Figure 3.9 and 3.10 represent the overall algorithm's process for a $(20, 7)$ punctured polar and PAC code with $w_{end} = 8$. The puncturing set is defined by

$$\mathcal{P} = \{0, 2, 4, 8, 10, 12, 16, 18, 20, 24, 26, 28\}$$

and the information set $\mathcal{I} = \{15, 22, 23, 27, 29, 30, 31\}$. We can see from Figure 3.9 that the overall RWEF in the case of punctured pure polar codes is:

$$A^8_{\mathcal{CP}}(X) = 1 + 2X^2 + X^4 + 14X^8 \tag{3.15}$$

This is justified by the fact that, at the last enumeration stage, i.e. when $i = s = 25$, $rk(GP_{26}^{31}) = 4$ and therefore $\frac{1}{2^{N-i-1-rk(GP_{s+1}^{N-1})}} = \frac{1}{4}$. This explains why the count of codewords for each weight is divided by 4.

The same principle holds for PAC codes. However, it's worth noting that for PAC codes, the minimum distance of the code increased from 2 in the case of pure polar codes to 6 for PAC codes. This indicates that the pre-transformation implemented for PAC codes raised the minimum distance of the code from 2 to 6.

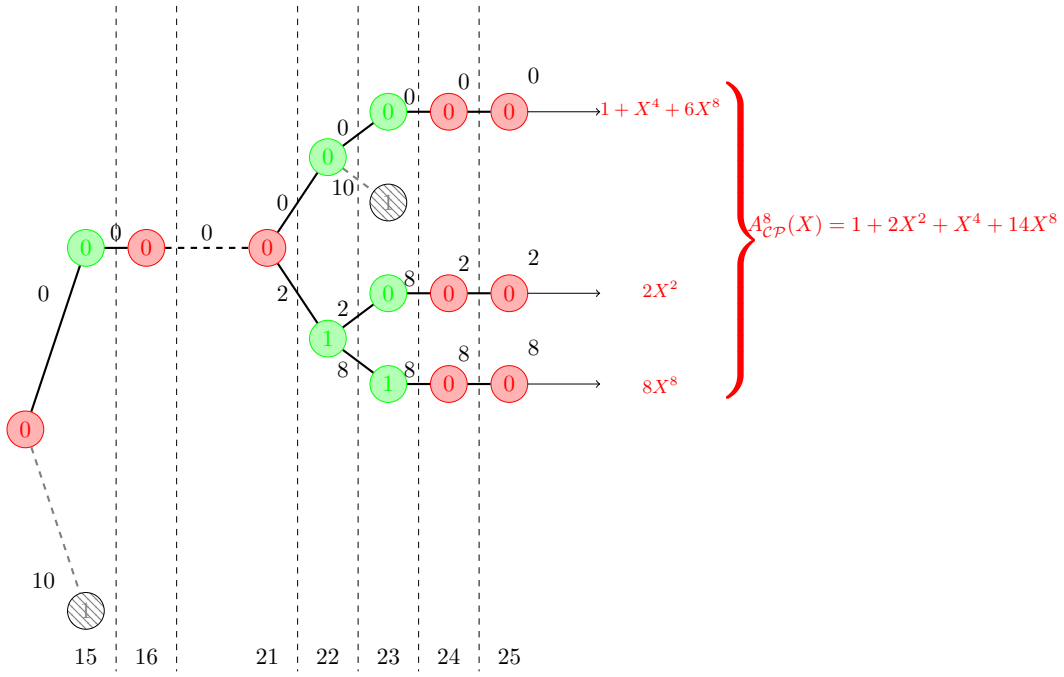


Figure 3.9: Relaxed reduced spectrum enumeration for a $(20, 7)$ punctured polar code

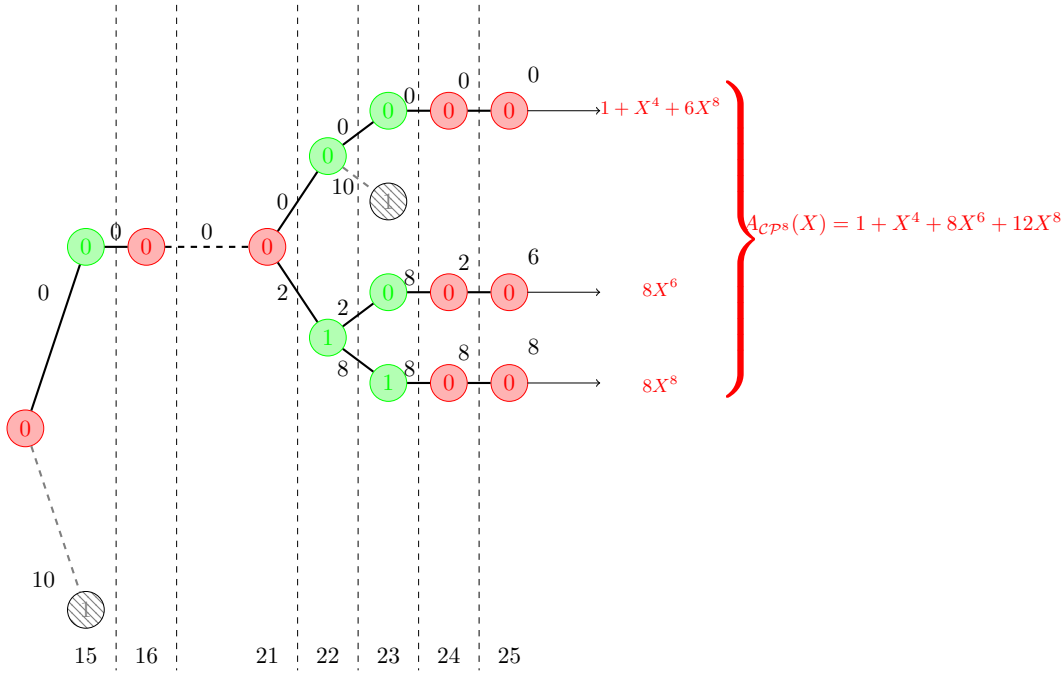


Figure 3.10: Relaxed reduced spectrum enumeration for for a $(20, 7)$ punctured PAC code

Algorithm 1: Enumeration of the low-weight codewords of punctured PAC codes

Input: Rate compatible punctured PAC code $\mathcal{CP}(N, K, \mathcal{F}, \mathbf{g}), \mathcal{P}, w_{end}$

Output: Reduced spectrum with weights less or equal to w_{end}

$L \leftarrow 1$
$\mathcal{L} \leftarrow \{0\}$ `/* List to store prefixes */`

for $i \in [\![0; s]\!]$ **do**
  for $l \in [\![1; L]\!]$ **do**
    if $i \in \mathcal{F}$ **then**
      for $l \in [\![1; L]\!]$ **do**
        $v_i[l] \leftarrow 0$
        $\hat{u}_i[l] = \mathbf{g}(\hat{\mathbf{v}}_0^i[l]))$
        Compute w^* of $\mathcal{CP}_N(\mathbf{u}_0^{i-1}[l], u_i[l])$
        Discard the prefixes for which $w^* \geq w_{end}$
        $L \leftarrow |\mathcal{L}|$
      end
    else
      $\mathcal{L} \leftarrow \mathcal{L} \bigcup \mathcal{L}'$ `/*` $\mathcal{L}'$ `is a copy of` $\mathcal{L}'$ `*/`

      for $l \in [\![1; L]\!]$ **do**
        $[v_i[l], v_i[l']] \leftarrow [0, 1]$
        $\hat{u}_i[l] = \mathbf{g}(\hat{\mathbf{v}}_0^i[l]))$
        $\hat{u}_i[l'] = \mathbf{g}(\hat{\mathbf{v}}_0^i[l']))$
        Compute w^* of $\mathcal{CP}_N(\mathbf{u}_0^{i-1}[l], u_i[l])$ and $\mathcal{CP}_N(\mathbf{u}_0^{i-1}[l'], u_i[l'])$
        Discard the cosets for which $w^* \geq w_{end}$
        $L \leftarrow |\mathcal{L}|$
      end
    end
  end
  if $i = s$ **then**
    Compute the RWEF of the remaining paths in the list
    Compute the RWEF of the overall code $A_{\mathcal{CP}}^{w_{end}}(X)$
end
Return $A_N^{w_{end}}(\mathcal{CP})$

3.7 Distance properties results for punctured and shortened polar and PAC codes

3.7.1 Minimum distance Properties

Table 3.1 lists the minimum distance d^*, the associated number of occurrences A^* as well as the total number of explored cosets n_r of punctured pure and pre-transformed polar codes. The code rates $R = 1/3$ and $R = 1/4$ are considered. For both tables, the

Algorithm 2: Enumeration of the low-weight codewords of shortened PAC codes

Input: Rate compatible Shortened PAC code $\mathcal{CS}(N, K, \mathcal{F}, \mathbf{g}), \mathcal{S}, w_{end}$

Output: Reduced spectrum with weights less or equal to w_{end}

$L \leftarrow 1$
$\mathcal{L} \leftarrow \{0\}$ `/* List to store prefixes */`

for $i \in [\![0; s]\!]$ **do**
  for $l \in [\![1; L]\!]$ **do**
    if $i \in \mathcal{F}$ **then**
      for $l \in [\![1; L]\!]$ **do**
        $v_i[l] \leftarrow 0$
        if $i \in \mathcal{S}$ **then**
          $u_i[l] \leftarrow 0$ `/* PAC constraint from (3.4). Only in the shortening case */`
        else
          $\hat{u}_i[l] = \mathbf{g}(\hat{\mathbf{v}}_0^i[l]))$
        end
        Compute w^* of $\mathcal{CS}_N(\mathbf{u}_0^{i-1}[l], u_i[l])$
        Discard the prefixes for which $w^* \geq w_{end}$
        $L \leftarrow |\mathcal{L}|$
      end
    else
      $\mathcal{L} \leftarrow \mathcal{L} \bigcup \mathcal{L}'$
      for $l \in [\![1; L]\!]$ **do**
        $[v_i[l], v_i[l']] \leftarrow [0, 1]$
        $\hat{u}_i[l] = \mathbf{g}(\hat{\mathbf{v}}_0^i[l]))$
        $\hat{u}_i[l'] = \mathbf{g}(\hat{\mathbf{v}}_0^i[l']))$
        Compute w^* of $\mathcal{CS}_N(\mathbf{u}_0^{i-1}[l], u_i[l])$ and $\mathcal{CV}_N^{(i)}(\mathbf{u}_0^{i-1}[l'], u_i[l'])$
        Discard the cosets for which $w^* \geq w_{end}$
        $L \leftarrow |\mathcal{L}|$
      end
    end
  end
  if $i = s$ **then**
    Compute the RWEF of the remaining paths in the list
    Compute the RWEF of the overall code $A_{\mathcal{CV}}^{w_{end}}(X)$
end
Return $A_N^{w_{end}}(\mathcal{CS})$

dynamic frozen bits are generated as a random combination of previous information bits. The frozen bit sets referred to as 5G are the ones specified in the 5G standard [9] and the ones referred to as 5G-RM are the ones introduced in [76]. Referring to Tables

3.2 and 3.1, the 5G-RM rate profile, when applicable, enhances the minimum distance of shortened polar codes. It is also observed, that both pre-transformations (PAC and DFB) reduce the number of codewords with minimum distance. The reduction is more significant in the case of the 5G-RM rate-profiling.

R	N	S	Code	Type	5G			5G-RM		
					d^*	A^*	n_r	d^*	A^*	n_r
1/2	128	48	(80,40)	Polar	8	1078	22116	-	-	-
				PAC	8	582	11789	-	-	-
				DFB	8	486	9554	-	-	-
	256	96	(160,80)	Polar	8	508	16933	16	104470	4659318
				PAC	8	300	10049	16	4166	51129
				DFB	8	348	11639	16	2920	19054
	512	192	(320,160)	Polar	8	120	6384	16	81500	1157676
				PAC	8	120	6384	16	22876	229005
				DFB	8	120	6384	16	17184	136029
3/4	128	48	(80,60)	Polar	4	764	25873	-	-	-
				PAC	4	636	21649	-	-	-
				DFB	4	620	21131	-	-	-
	256	96	(160,120)	Polar	4	120	6208	8	81820	1550796
				PAC	4	120	6208	8	21116	207530
				DFB	4	120	6208	8	17179	132831
	512	192	(320,240)	Polar	8	110584	1282001	-	-	-
				PAC	8	73784	565874	-	-	-
				DFB	8	69392	515953	-	-	-

TABLE 3.1: Minimum distance properties of pure, PAC and DFB shortened polar codes

3.7.2 Reduced weight spectrum

This section summarizes the experimental results obtained on the partial weight distribution for a wide range of pure and pre-transformed rate-compatible polar codes. For each code, we compute the exact number A_w of codewords of weight w for all $\leq w_{end}$.
Table 3.3 gives the partial weight spectrum of punctured shortened polar and PAC codes for $N = \{128, 256, 512, 1024\}$, $P = \{48, 96, 192, 384\}$ and $S = \{48, 96, 192, 384\}$, respectively. We apply puncturing for the codes with code rate $R = 0.25$ and shortening for codes with code rate $R = 0.5$. This choice aligns with the 5G standardization, where shortening is used for high rates and puncturing for low rates [9]. In the case of PAC codes, the polynomial $\mathbf{g} = [1, 0, 1, 1, 0, 1, 1]$ is chosen. The frozen bit sets are the ones specified in the 5G standard [9].
The results for the number of codewords with minimum weight of shortened polar and

R	N	P	Code	Type	5G		
					d^*	A^*	n_r
1/4	128	48	$(80, 20)$	Polar	8	30	160
				PAC	8	2	61
				DFB	8	2	67
	256	96	$(160, 40)$	Polar	8	12	131
				PAC	8	8	130
				DFB	8	12	131
	512	192	$(320, 80)$	Polar	16	476	2923
				PAC	16	76	391
				DFB	8	100	471
1/3	128	48	$(80, 27)$	Polar	8	38	127
				PAC	8	14	65
				DFB	4	14	67
	256	96	$(160, 53)$	Polar	8	76	571
				PAC	8	8	200
				DFB	8	20	236
	512	192	$(320, 107)$	Polar	8	24	395
				PAC	8	12	392
				DFB	8	16	393

TABLE 3.2: Minimum distance properties of pure, PAC and DFB punctured polar codes

PAC codes (results highlighted in red) were corroborated with results in [76]. To the best of the author's knowledge, the full results for the partial weight spectrum of PAC codes have not been reported in the literature. As shown in Table 3.3, PAC codes have fewer low-weight codewords.

Moreover, a computational complexity comparison of the proposed algorithm to the one introduced in [14] is provided in Table 3.4. To this end, the punctured and shortened patterns were defined randomly to accommodate with the results given in [14]. This is made possible by the ability of Algorithm 1 to be adapted to any puncturing/shortening pattern. The results are shown for a $(200, 100)$ polar code with the same rate-profiling. We showed in [77] that the computation of the minimum weight of a coset has a worst-case computational complexity equal to $6(N-1)$. Algorithm 1 has therefore a worst case computational complexity equal to $6n_c(N-1)$, whereas the algorithm proposed in [14] has a computational complexity dominated by the term $\Sigma\mathcal{N}_{Subcode}^{(\mathcal{X})}$. It is shown in Table 3.4 that the number of codewords with a specific weight obtained via Algorithm 1 is in the same range of the results computed in [14]. Note that since the puncturing and shortening are done randomly, we cannot reproduce exactly the same results. Table 3.4 shows that the computational complexity of Algorithm 1 is lower by several orders of magnitude. For instance , it is indicated in [14] that the overall running time for a C++ implementation on a computer with 6 cores i7 and a 3.2GHz processor in the case

of a randomly shortened $(200, 100)$ polar code is approximately 28 hours. In contrast, our MATLAB implementation on a computer with 2 cores i5 and a 3.1GHz processor achieves a running time of less than 10 minutes.

(N, K)	Type	(w, A_w)
$(80, 20)$	Polar	$(8, 30), (16, 173), (20, 256)(24, 8040)$ $(28, 7424)$
	PAC	$(8, 2), (12, 8), (14, 8), (16, 109), (18, 56),$ $(20, 920), (22, 504), (24, 2456)$ $(26, 5352), (28, 11528), (30, 11304)$ $(32, 34194)$
$(160, 40)$	Polar	$(8, 12), (16, 382), (24, 2220), (32, 55533),$ $(40, 663536)$
	PAC	$(8, 8), (16, 230), (18, 64), (20, 64),$ $(22, 64), (24, 2120), (26, 960)$ $(28, 1216), (30, 1472), (32, 16557),$ $(34, 16448), (36, 19264)$ $(38, 22080), (40, 286240)$
$(320, 80)$	Polar	$(16, 476), (24, 11584), (28, 12288),$ $(32, 117598), (36, 12288)$ $(40, 678208), (44, 589824),$ $(48, 15764476)$
	PAC	$(16, 76), (18, 16), (20, 32), (22, 16),$ $(24, 208), (26, 112), (28, 352), (30, 112)$ $(32, 9694), (34, 2928), (36, 12512),$ $(38, 8176), (40, 160848), (42, 52496)$ $(44, 224544), (46, 192784), (48, 3669484)$
$(640, 160)$	Polar	$(16, 344), (24, 2688), (32, 117004)$ $(40, 3741824)$
	PAC	$(16, 20), (24, 24), (32, 11652), (34, 64)$ $(36, 704), (38, 64), (40, 47032)$
$(80, 40)$	Polar	$(8, 1078), (12, 32128), (14, 45056)$ $(16, 971821), (18, 2191360)$ $(20, 35615872)$
	PAC	$(8, 582), (10, 608), (12, 14848)$ $(14, 46624), (16, 446125), (18, 1810400)$ $(20, 11718144)$
$(160, 80)$	Polar	$(8, 508), (12, 2496), (16, 320030)$ $(20, 9821632)$
	PAC	$(8, 300), (12, 2112), (16, 92862)$ $(18, 15616), (20, 2453568)$
$(320, 160)$	Polar	$(8, 120), (16, 183116), (20, 731136)$
	PAC	$(8, 120), (16, 74540), (18, 6080)$ $(20, 568832)$
$(640, 320)$	Polar	$(16, 69496), (24, 27166592)$
	PAC	$(16, 43544), (18, 736), (20, 25408)$ $(22, 6208), (24, 16013568)$

TABLE 3.3: Partial weight distribution of punctured and shortened polar and PAC codes

(N, K)	$(200, 100)$ Random Shortening	$(200, 100)$ Random puncturing
(w, A_w)	$(8, 194)$	$(7, 12)$
	$(12, 456)$	$(8, 35)$
	$(16, 67867)$	$(9, 115)$
	$(20, 1319413)$	$(10, 332)$
	$(22, 696208)$	$(11, 710)$
		$(12, 1349)$
		$(13, 2934)$
		$(14, 6737)$
		$(15, 16490)$
		$(16, 41033)$
		$(17, 98835)$
		$(18, 235252)$
		$(19, 561588)$
$\Sigma\mathcal{N}^{(\mathcal{X})}_{Subcode}$ [14]	56257×10^9	128664×10^9
$TC_{w_{end}}$	78×10^9	48×10^9

TABLE 3.4: Partial weight distribution of $(200, 100)$ randomly punctured / shortened polar codes

3.8 Conclusion

In this chapter, we expanded upon the contributions of Chapter 2 to enable the computation of the number of codewords with minimum or low weight for punctured and shortened pure and pre-transformed polar codes. We introduced the concept of rate-compatible polar cosets to account for the effects of puncturing and shortening. Subsequently, we demonstrated that, similarly to polar cosets, the distance properties of rate-compatible polar cosets can be efficiently determined thanks to factor graphs. This paved the way for the development of a generic low-complexity algorithm for computing the minimum distance properties, or more generally, the reduced weight spectrum of punctured and shortened pure and pre-transformed polar codes.

Our experimental results underscored the significantly reduced computational complexity of the proposed algorithm compared to existing approaches. Additionally, we illustrated that, under specific rate-profile configurations, pre-transformed shortened/punctured polar codes demonstrate enhanced performance thanks to the improvements of their distance properties.

The next chapter combines the insights gained from the current and preceding chapters in order to provide an analysis on the different parameters that enables code constructions offering a trade-off between distance properties and decoding computational complexity.

4 Towards a trade-off between distance properties and SCL decoding complexity of polar codes

This chapter begins by introducing a method to optimize the rate-profile construction of polar codes, focusing on their distance properties. Since polar codes designed solely with distance constraints perform poorly under SCL decoding with a moderate list size, we investigate the parameters that influence the average list size required to achieve Maximum-Likelihood performance. Additionally, we explain how certain paths can be pruned during the list decoding process without impacting the overall performance.

4.1 Introduction

Polar codes performance under near-ML decoding is essentially dependant to their weight spectrum and more specifically to their number of low weight codewords. The advantage

behind leveraging distance properties in code definition lies in their ability to offer constructions for which the ML bound is near to the achievability bound. Consequently, it makes their performance under near-ML decoders such as the Fano decoding very appealing. In the previous chapters, we introduced a streamlined approach for computing the partial weight spectrum of various types of polar codes: pure, pre-transformed, punctured and shortened. This method is one of the efficient pathways towards the design of codes based on their distance properties. In previous works, such as [78], Monte Carlo simulations were used to identify good codes under Fano decoding. In [42], PAC code constructions are introduced based on a genetic algorithm to optimize distance properties. Both [42] and [78] proposed codes that perform close to the RCU bound when decoded under Fano decoding. These constructions hold particular appeal, especially for short-length codes where achieving the ML performance entails only moderate decoding computational complexity. However, both of these methods are limited by their high decoding computational complexity for higher block-lengths. For moderate to high block lengths and using SCL decoding with a moderate list size, both pure and precoded polar codes tend to perform poorly. This has prompted extensive research aimed at developing polar and precoded polar codes that achieve a balance between strong distance properties and effective performance with short to moderate list sizes. It can also be considered to adapt the decoders in order to reduce the overall computational complexity. Three primary approaches were pursued in the state of the art:

1. The first approach involves constructing codes that strike a balance between their distance properties, which influence their ML bounds, and the computational complexity of decoding process. Several methods have been explored in this regard. Firstly, [44] introduced constructions similar to RM-polar, considering both the distance properties of the code and the reliability of information bits. While demonstrating improved performance under short list decoding, this construction relies on Monte Carlo simulations, resulting in high computational complexity. Similarly, [43] proposed a bit-channel selection algorithm that considers both the effect of the polarization effect and the ML decoding performance as selection criteria. In [79], a genetic algorithm was used to design rate-profile tailored to a specific decoding algorithm. [80] put forth a recursive algorithm for the rate-profiling polar codes with dynamic frozen bits, also emphasizing both distance properties and performance under SCL decoding with moderate list sizes. In [45], a novel approach is proposed based on the identification information-theoretic quantities associated with the required list size, on average, to achieve ML performance, thereby proposing rate profiles that outperform standard polar codes. This approach was further revised in [46], where a genetic algorithm was applied to develop new rate-profile

strategies for polar codes with dynamic frozen bits. [81] devised a search-constrained optimization method to evaluate the performance of PAC codes under Fano decoding. It enables to offer performance near the RCU bound while reducing the Average Number of Visits (ANV) of the Fano decoder. These methodologies collectively aim to strike a balance between code performance and computational complexity. Thereby, they address the limitations associated with decoding algorithms for polar codes.

2. Another approach involves the optimization of the various precoding formats in order to improve the distance properties of polar codes. For example, [51] proposed optimizing the CRC for polar codes for specific block lengths and code rates. This optimization yields concatenated polar codes with CRC that exhibit better performance compared to those employing conventional CRC polynomials. Additionally, [12] introduced a precoding technique known as "row merging" to substitute the typical rate-1 precoding used in PAC codes, while either maintaining or enhancing the distance properties of polar codes. Expanding upon this method, [82] broadened the scope by constructing row-merged polar codes with increased code rates without compromising their distance properties. Furthermore, [83] employed a greedy strategy to fine-tune row-merged polar codes, resulting in codes with even stronger distance properties.

3. A last way is to reduce the size of the list whenever it is relevant during the decoding process. In this context, [84] and [85] introduced criteria following which either SC or SCL are performed on a decoding step. Other methods, such as [86] essentially relied on the pruning of paths for which the metric value is greater than a threshold. [87] introduced a *path metric range* that allows for the reduction of the list size when it takes large values. Some of these approaches can significantly reduce the average list size for a moderate performance loss but, as any dynamic method, it suffers from a worst case complexity that is actually equivalent (if not worse) to the complexity of the SCL.

This chapter shifts focus to the different strategies that aim to construct codes with good distance properties and/or reduce the decoding computational complexity. In particular, we present:

1. A greedy algorithm for the design of PAC polar codes based on their distance properties (This approach can also be generalised to polar codes with other pre-transformation cases) . This algorithm relies on the contributions presented in the previous chapters and offers code constructions that perform near the achievability

bound for short-length PAC codes under SCL decoding with moderate list size.

2. A pruning technique for both SCL and CA-SCL decoders that enables, from a certain index to switch from list decoding to SC decoding while not altering the performance.

3. An upper bound on the average list size required to achieve ML performance under SCL decoding process. This estimation is based on path retention probabilities at each decoding step and enables to have an idea on the computational complexity of a near-ML decoding of polar codes. Unlike the bounds presented in [45], this approach takes into account the impact of pre-transformation. Moreover, it is adaptable to punctured and shortened pure and pre-transformed polar codes.

4.2 Distance properties based rate-profile for PAC codes

We present in this section a rate-profile construction for PAC codes. Note that this method is presented for PAC codes but can also be extented to any pre-transformed polar code. This construction is based on the results of Chapter 2 that enable the efficient computation of the distance properties of PAC codes. In order to simplify the optimization matter, we mainly focus on the minimum distance properties i.e. the minimum distance and the number of codewords with minimum distance. First, we outline the overall procedure of the algorithm. Following this, we introduce a simplification to reduce its computational complexity.
The goal of rate-profile construction is to choose K information bits from a total of N available bits. We propose a greedy method that selects information bits incrementally until K bits have been chosen. At each step, the selected information bit is the one that either maximizes the minimum distance or minimizes the number of codewords with the minimum weight. This allows an order on the bits following their distance properties. To do so, this rate-profile construction relies on the RM scores assigned to each row of the generator matrix. Initially, the last bit $N-1$, i.e., the bit associated to the last row of the generator matrix, has the highest RM score. Therefore, it is the first one appended to the information set. Then, during each iteration for the selection of the next information bit, the bit associated to the row with the highest remaining RM score and maximizing the minimum distance and/or minimizing the number of codewords with minimum weights is selected at a time. This process is repeated until ordering all the bits.

Example 4.2.1. *The process of the rate-profile construction is represented in Figure 4.1 for the* 6^{th} *and* 7^{th} *step for a PAC code with* $N = 16$ *and* $\mathbf{g} = [1, 0, 1, 1, 0, 1, 1]$. *Starting*

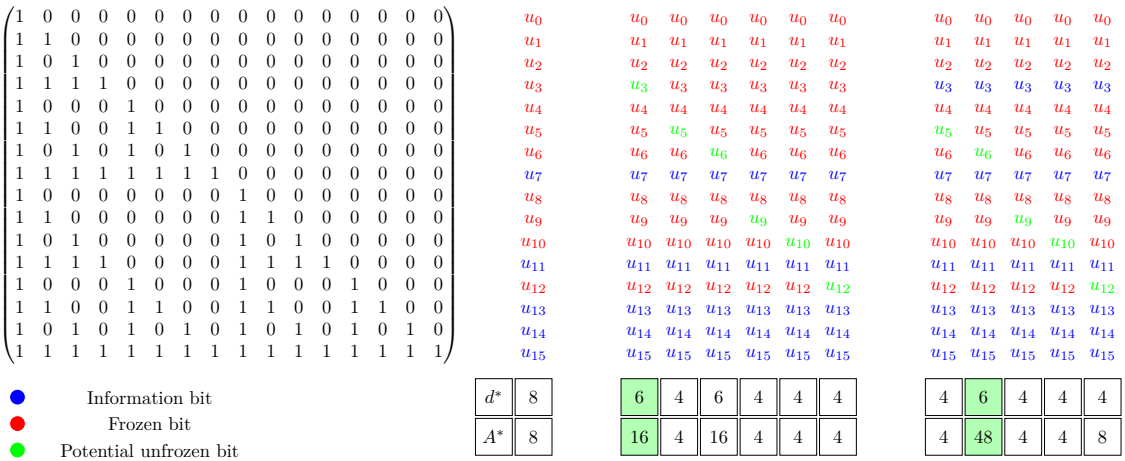


Figure 4.1: $(16, 7)$ proposed rate-profiling for a PAC code

from a $(16, 5)$ *PAC codes, all the bits with an associated row weight greater or equal to 8 have been selected. Therefore, we only consider the bits with row weights equal to* 4*, i.e.* u_3*,* u_5*,* u_6*,* u_9*,* u_{10} *or* u_{12}*. To determine the next information bit position, we iterate through the bits one by one, unfreezing each one and evaluating the resulting distance properties. By computing the minimum Hamming distance* d^* *and the associated number of occurrences* A^* *for each configuration, we identify the setup that offers the best distance properties. During this process, we initially unfreeze bit* u_3*, followed by bit* u_6*, as these steps lead to the most favorable rate-profile constructions in terms of distance properties.*

Remark 1: Note that this process guarantees the largest minimum distance in the case of pure polar codes. But it may not be true in the case of pre-transformed polar codes. Actually, since the pre-transformation can enhance the code's minimum distance, specific configurations can have a minimum distance d^* while having information bits whose associated weight rows are lower than d^*. This is the case in [42] where the presented $(128, 64)$ PAC code has a minimum distance of 16 while having information bits whose associated row weights are equal to 8. Therefore, the proposed approach in this section only provides a near optimal construction.

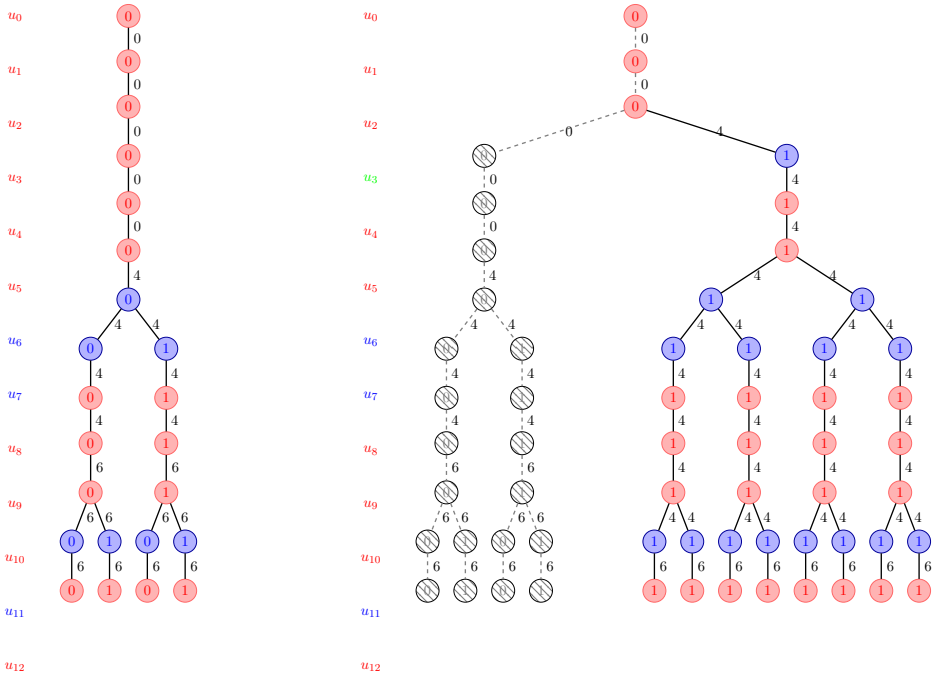
Remark 2: The considered rate-profile construction is based on a greedy algorithm that during each iteration selects the information bits that maximizes the minimum distance or/and minimizes the number of codewords with minimum weight. This method is sub-optimal as the optimal construction of a (N, K) pure or pre-transformed polar code would require considering all the possible combinations of K bits. Considering the example of the $(64, 32)$ polar code, 32 information bits have to be chosen among 64 bits. In that case, the number of possible combinations is equal to $C_{32}^{64} \approx 1832624140942590534$

which makes considering this approach infeasible.

4.2.1 Overall algorithm

In order to compute d^* and A^* for each configuration, the algorithms proposed in Chapter 2 are applied. However, a simplification can be introduced in order to further reduce the computational complexity. Starting with an example, we show how the simplification operates.

Example 4.2.2. *Figures 4.2a and 4.2b represent the enumeration process in order to determine d^* and A^* detailed in Algorithm 2 in Chapter 2 for $(16, 6)$ and $(16, 7)$ PAC codes with $\mathbf{g} = [1, 0, 1, 1, 0, 1, 1]$.*



(a) $(16, 6)$ PAC code (b) $(16, 7)$ PAC code

Figure 4.2: Determination of d^* and A^* for a $(16, 6)$ and $(16, 7)$ PAC code

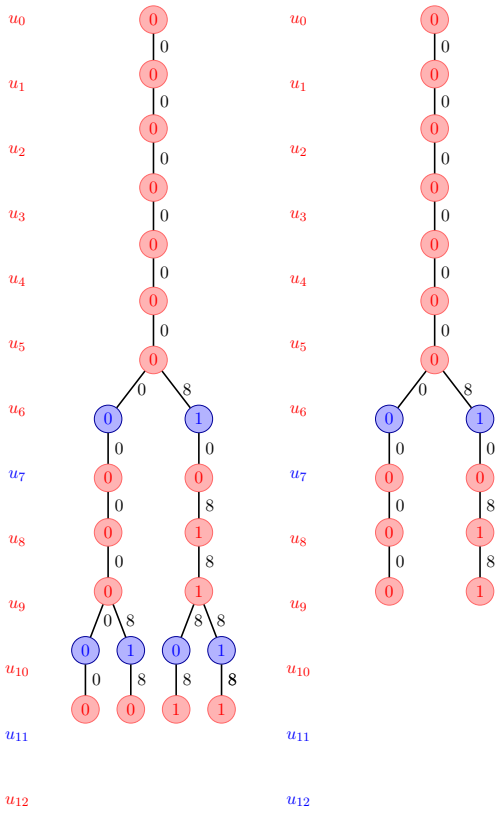
In contrast to the $(16, 6)$ configuration, the $(16, 7)$ setup involves the bit u_3 being unfrozen. This implies that both the cosets where $u_3 = 0$ and $u_3 = 1$ have to be evaluated. Since the case where $u_3 = 0$ has already been explored in the $(16, 6)$ configuration, there is no need to process it again. Since the minimum distance and the number of codewords

with minimum weight has been computed for the $(16, 6)$*, we only have to consider the case where* $u_3 = 1$ *for the* $(16, 7)$ *configuration. At the end of the enumeration, if the minimum weights of the* $(16, 6)$ *configuration and the* $(16, 7)$ *configuration are equal, which is verified in this case, the total number of codewords with minimum weight is obtained as the sum of the number of codewords with minimum weight computed for the* $(16, 6)$ *code and the number of codewords with minimum weight computed for the* $(16, 7)$ *code without considering* $u_3 = 0$*. This simplification avoids re-computing the dashed graph in Figure 4.2a that has been computed during the previous step. However, it should be noted that the computational complexity reduction highly depends on the position of the new unfrozen bit. In Figure 4.2, the number of avoided computations corresponds to half of the total number of computations. It is the maximum number of computations that can be removed. In other cases, the number of avoided computations might be less or even equal to* 0*. Thus, a case where* u_{12} *is unfrozen is represented in Figure 4.3b. We can observe that, the last frozen is no longer* u_{12} *but is* u_{10}*. It means that, all the previous steps have to be re-computed.*

Algorithm 1 contains the details of the considered approach. The main steps can be described as follows:

- a set of indexes is defined. This set is composed with the indexes whose associated row weights are equal to the highest remaining RM score.
- From this set, one bit is unfrozen at a time and the minimum distance and the number of codewords with minimum weight are computed. This computation is based on the approach explained in Example 4.2.2. Thus, only the paths resulting from unfreezing the considered bit are considered and the overall minimum distance and the number of codewords with minimum distance are determined by combining this result and the previous computations of d^* and A^*
- The configuration that guarantees the highest minimum distance and/or the lowest number of codewords with minimum weights is chosen and the information set is updated accordingly.
- The process is iterated until ordering all the channels.

This approach offers a significant advantage by generating a channel order based on the minimum distance characteristics inherent to the polar code under consideration, unlike the methodology proposed in [42], where an execution of the algorithm is necessary for every specific code rate. Moreover, no genetic algorithm with a costly iterative process has to be applied.



(a) $(16, 5)$ PAC code (b) $(16, 6)$ PAC code

Figure 4.3: Determination of d^* and A^* for a $(16, 5)$ and $(16, 6)$ PAC codes

Algorithm 1: Channels order based on distance properties

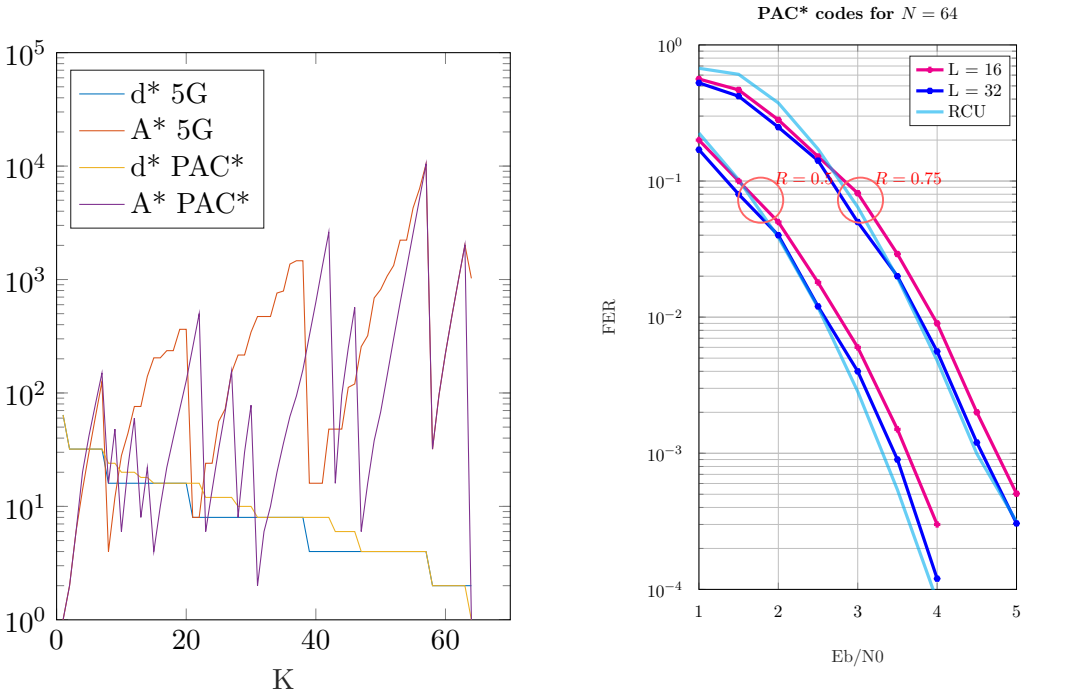
Input: $N, \mathbf{g}$

Output: Set of ordered channels $\mathcal{I}$ according to distance properties

$\mathcal{I} \leftarrow \{N-1\}$ $d^* \leftarrow N$, $A^* \leftarrow 1$, $K \leftarrow 1$, $s \leftarrow N-2$
while $K < N$ **do**
  $\mathcal{I}_p \leftarrow$ `SelectRMIndexes(`N, K`)`
  for $i \in \mathcal{I}_p$ **do**
    $\mathcal{I} \leftarrow \mathcal{I} \bigcup \{i\}$
    $d_{temp} \leftarrow 0$, $A_{temp} \leftarrow +\infty$
    $[d_i, A_i, s_i] \leftarrow$ `ComputeAD(`$N, \mathcal{I}, \mathbf{g}, i, s, d^*, A^*$`)`
    if $d_i > d_{temp}$ **then**
      $d_{temp} \leftarrow d_i$, $A_{temp} \leftarrow A_i$, $i_{opt} \leftarrow i$
    else if $d_i = d_{temp}$ **then**
      if $A_i < A_{temp}$ **then**
        $A_{temp} \leftarrow A_i$, $i_{opt} \leftarrow i$
    end
    $\mathcal{I} \leftarrow \mathcal{I} \backslash \{i\}$
  end
  $d^* \leftarrow d_{temp}$, $A^* \leftarrow A_{temp}$, $\mathcal{I} \leftarrow \mathcal{I} \bigcup \{i_{opt}\}$
end
return $\mathcal{R}$
subroutine `ComputeAD(`$N, \mathcal{I}, \mathbf{g}, i_{prev}, s_{prev}, d_{prev}, A_{prev}$`)`
  $F \leftarrow \bar{\mathcal{I}}$, $s \leftarrow max(F)$
  if $s \neq s_{prev}$ **then**
    Perform Algorithm 2 in Chapter 2
    return (d^*, A^*)
  else
    for $i \in [\![0, s]\!]$ **do**
      if $i \neq i_{prev}$ **then**
        Perform the same steps of Algorithm 2 in Chapter 2
      else
        for $l \in [\![1; L]\!]$ **do**
          $[u_i[l], s_{i+1}[l]] \leftarrow \text{conv}(1, s_i[l])$
          Compute w^* and A^* and discard the paths for which $w^* > d_{prev}$
        end
      end
    end
    if $d^* = d_{prev}$ **then**
      return $(d^*, A^* + A_{prev})$
    else
      return (d^*, A^*)
    end
  end

4.2.2 Rate-profile construction results

We present in this section the rate-profiling constructions obtained for different code-lengths in the case of PAC codes. In all the following, we denote by PAC*, PAC codes with rate-profiles obtained using Algorithm 1. Their distance properties and the performance of code under this construction are then discussed.
Figure 4.4a represents the different minimum distances and associated number of occurrences for a PAC* codes with $N = 64$ and $\mathbf{g} = [1, 0, 1, 1, 0, 1, 1]$ using Algorithm 1 and 5G rate-profiling. We can see from this figure that the minimum distance is enhanced and/or the number of codewords with minimum distance is reduced. For instance, a $(64, 8)$ PAC code has a minimum distance of 16 under 5G construction, whereas it has a minimum distance equal to 24 under the construction based on Algorithm 1.
In order to evaluate the performance of short codes with a rate-profile built following



(a) Evolution of d^* and A^* for a PAC code with $N = 64$ under Algorithm 1 and 5G construction

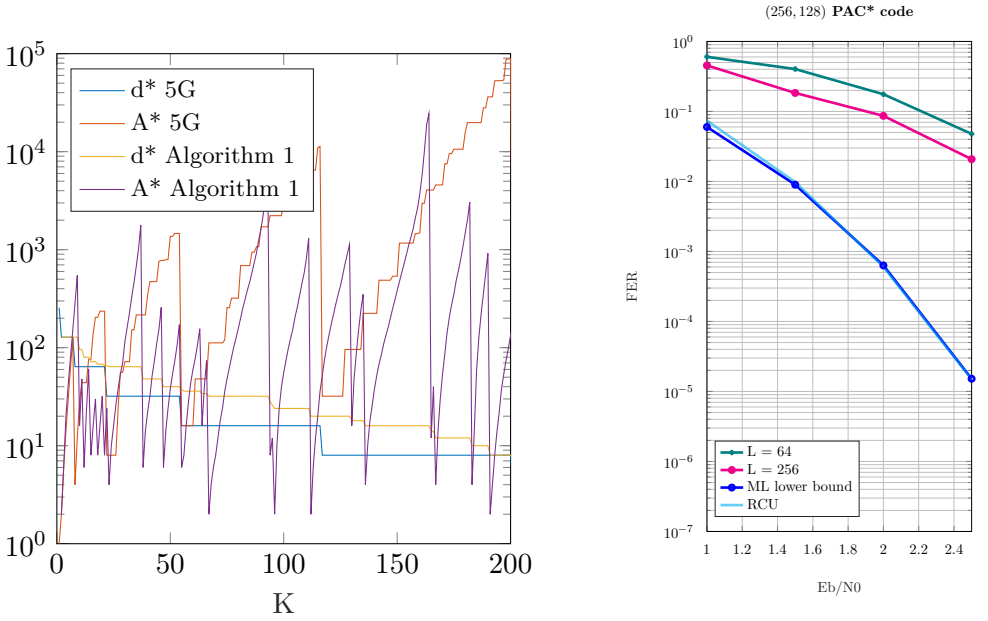
(b) Performance of PAC* codes with $N = 64$ and $K = \{16, 32\}$ under list decoding with $L = \{16, 32\}$

Figure 4.4: Construction of PAC* rate-profile and performance evaluation for $N = 64$

the proposed approach, Figure 4.4 shows the performance of a PAC* code with $N = 64$ for different code rates and under SCL with different list sizes. The performance results are compared to RCU bound. Under the rate-profile construction described in Algorithm

1, both $(64, 32)$ and $(64, 48)$ PAC* codes yield similar performance to the RCU bounds with SCL decoding with list size $L = 32$. This shows that this method enables to built rate-profiles that yield performance very close to RCU bound. Moreover, it is also observed that a small list size ($L = 32$) enables to achieve the maximum likelihood performance and that even SCL decoding with a list size $L = 16$ performs closely to the RCU bound. While the $(64, 32)$ PAC codes in [44] and [78] have $d^* = 8$ and $A^* = 8$, in our case, $d^* = 8$ and $A^* = 6$. In the case of the $(64, 48)$ PAC* code, $d^* = 4$ and $A^* = 16$. The $(64, 48)$ PAC code under 5G rate-profiling has $d^* = 4$ and $A^* = 320$.
The evolution of d^* and A^* are represented in Figure 4.5a for a PAC* codes with $N = 256$ and $\mathbf{g} = [1, 0, 1, 1, 0, 1, 1]$.

The same observations for the PAC* codes with $N = 256$ can be made. In fact, the rate-profiling introduced in Algorithm 1 enhances the minimum distance and/or reduces the number of codewords with minimum weight compared to a 5G rate-profiling. Figure 4.5b shows the performance of a $(256, 128)$ PAC* code under list sizes 64 and 256. However,



(a) Evolution of d^* and A^* for a PAC code with $N = 256$ under Algorithm 1 and 5G construction

(b) Performance of PAC* codes with $N = 256$ and $K = 128$ under list decoding with $L = \{64, 256\}$

Figure 4.5: Construction of PAC* rate-profile and performance evaluation for $N = 256$

on these cases, the SCL decoder have poor performance even under list decoding with list size $L = 256$. This observation serves as a catalyst for the investigations detailed in subsequent sections. This pursuit is motivated by the inadequacy of distance-based

rate-profile constructions under list decoding with moderate list sizes, which deliver sub-optimal results for medium-length codes. The challenge lies in identifying the decoding steps at which a larger list size is necessary to approximate ML-decoding performance for a given rate profile. In the next section, we introduce an initial approach that allows switching from SCL decoding to SC decoding at a certain step in the decoding process, while maintaining the same performance as SCL or CA-SCL decoding.

4.3 Tailored list decoding of polar codes

4.3.1 Decoding tail in SC-based algorithms

It has been observed in [8] that a genie-assisted SC algorithm drastically improves the decoding performance. In such an approach, a "genie" is applied to correct up to two errors during the SC decoding process. In other words, during SC-based decoding, the very first errors have a major impact on the decoding performance. In Figure 4.6, the distribution of the first error positions of the SC decoder is provided for 1000 sequences that the SC decoding process failed to correctly decode from a $(256, 128)$ polar code. This Monte Carlo simulation shows that the end of the decoding sequence, here denoted as "tail", is almost error-free. This means that applying list-decoding at the end of the sequence is superfluous. Actually, a simple SC decoder would provide very similar performance. Based on this observation, we propose a simplification of the SCL and the CA-SCL decoding algorithms in which the "tail" of the sequence is decoded with SC decoding. More precisely, starting from a certain index $\omega = N - T$, the path metrics are no longer updated nor is the metric sorting process. Instead, the L paths are kept and an SC decoding is performed independently on each one of these paths. At the end of the decoding, the path with the lowest path metric is selected. The determination of the index ω depends on the considered decoder (SCL or CA-SCL).

4.3.2 Tailoring in the case of SCL decoding

Let us denote by γ the number of information bits before the last frozen bit s of a polar code. It has been proven in both [45] and [88] that an SCL decoder with a list size equal to 2^{γ} achieves Maximum Likelihood decoder performance. In this case, γ denotes the previously defined mixing factor, i.e. the number of information bits before the last frozen bit. it is explained in [88] that this implies that after the last frozen bit s, the SCL decoding can be replaced with a decoder that selects the path that has the closest distance to the received codeword. In other words, the SCL decoder is performed on the first s bits. Then at the s^{th} decoding step, the proposed method is applied to compute

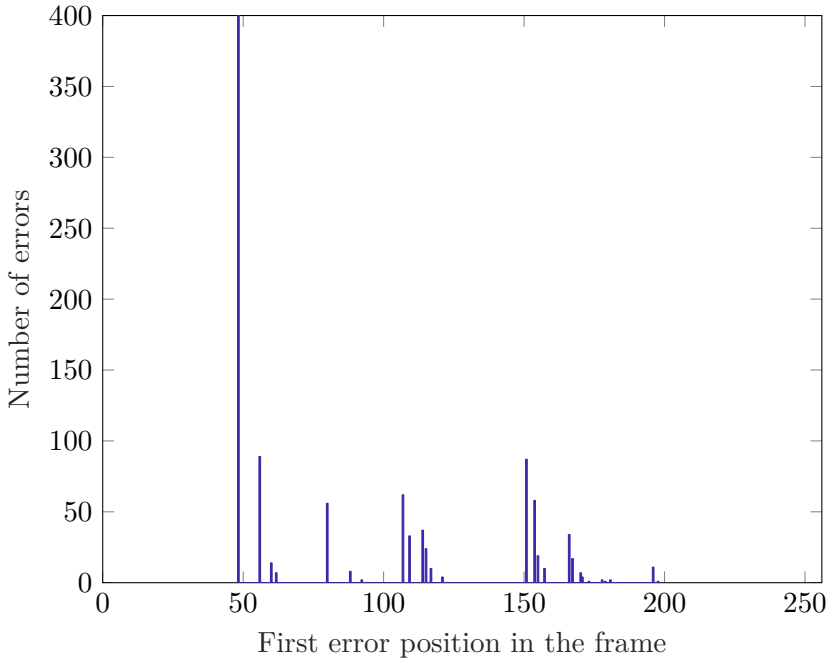


Figure 4.6: First error position histogram for a (256,128) polar code decoded with an SC decoding algorithm at $\frac{E_b}{N_0} = 3$ dB

the distance of the received codeword to all the cosets generated by the prefixes in the list. Only the coset with the minimum distance is kept then an SC decoding is performed on this particular coset.

In this section, we leverage this result to demonstrate that transitioning from Successive Cancellation List (SCL) decoding to Successive Cancellation (SC) decoding after the last frozen bit s does not impact decoding performance for a list decoding with an arbitrary list size L. Furthermore, we establish that the coset with the minimum distance from the received codeword corresponds to the coset with the lowest metric. This assertion is valid for pure and precoded polar codes.

4.3.2.1 SCL decoding metrics

In this section, we express the path metrics of SCL decoding in terms of the distance between the received codeword and the cosets within the list. This explains why SCL decoding becomes unnecessary after processing the final frozen bit.
It has been shown in [26] that given the l^{th} prefix remaining in the list $\mathbf{u}_0^i[l]$, the path

metric $m_i(\mathbf{u}_0^i[l], \mathbf{y})$ can be expressed as:

$$m_i(\mathbf{u}_0^i[l], \mathbf{y}) = -\log\left(\mathbb{P}\left(\mathbf{U}_0^i = \mathbf{u}_0^i[l] | \mathbf{Y} = \mathbf{y}\right)\right) \tag{4.1}$$

$m_i(\mathbf{u}_0^i[l], \mathbf{y})$ can also be expressed as :

$$m_i(\mathbf{u}_0^i[l], \mathbf{y}) = \frac{1}{2\sigma^2} \min_{\mathbf{x} \in \mathcal{C}_N(\mathbf{u}_0^i[l])} \left(||\mathbf{y} - \mathbf{x}||_2^2\right) - \frac{1}{2\sigma^2} \min_{\mathbf{v} \in \{-1,1\}^n} \left(||\mathbf{y} - \mathbf{v}||_2^2\right) \tag{4.2}$$

The detailed proof in given in Appendix C The path metric of a prefix $\mathbf{u}_0^i[l]$ is formed by two terms:

- The term $\frac{1}{2\sigma^2} \min_{\mathbf{x} \in \mathcal{C}_N(\mathbf{u}_0^i[l])} \left(||\mathbf{y} - \mathbf{x}||_2^2\right)$ represents the minimum Euclidean distance of the received codeword to the coset generated by $\mathbf{u}_0^i[l]$.
- The term $\frac{1}{2\sigma^2} \min_{\mathbf{v} \in \{-1,1\}^n} \left(||\mathbf{y} - \mathbf{v}||_2^2\right)$ represents the minimum Euclidean distance of the received codeword to any codeword in $\mathbb{F}_2^N$. This term is common to all the metrics $m_i(\mathbf{u}_0^i[l])$.

Therefore the path metrics at a specific step i during SCL decoding process directly illustrates the minimum Euclidean distance between the received codeword $\mathbf{y}$ and the cosets generated by $\mathbf{u}_0^i[l], \forall l \in [1, L]$. In the rest of this analysis, we only focus on the term $\frac{1}{2\sigma^2} \min_{\mathbf{x} \in \mathcal{C}_N(\mathbf{u}_0^i[l])} \left(||\mathbf{y} - \mathbf{x}||_2^2\right)$ since the other term is common to all paths.
Let us consider SCL decoding with a list size L at the s^{th} step (i.e. last frozen bit's step). We denote by $\mathbf{u}_0^s[l_{min}]$ the path in the list with the least metric $m_s(\mathbf{u}_0^s[l_{min}])$.

$$m_s(\mathbf{u}_0^s[l_{min}]) = \frac{1}{2\sigma^2} \min_{\mathbf{x} \in \mathcal{C}_N(\mathbf{u}_0^s[l_{min}])} \left(||\mathbf{y} - \mathbf{x}||_2^2\right) - \frac{1}{2\sigma^2} \min_{\mathbf{v} \in \{-1,1\}^n} \left(||\mathbf{y} - \mathbf{v}||_2^2\right)$$

Since $\mathcal{C}_N(\mathbf{u}_0^s[l_{min}]) \subset \mathcal{C}$, there exists $\mathbf{x}_{min} \in \mathcal{C}_N(\mathbf{u}_0^s[l_{min}])$ such that:

$$m_s(\mathbf{u}_0^s[l_{min}]) = \frac{1}{2\sigma^2} ||\mathbf{y} - \mathbf{x}_{min}||_2^2 - \frac{1}{2\sigma^2} \min_{\mathbf{v} \in \{-1,1\}^n} \left(||\mathbf{y} - \mathbf{v}||_2^2\right)$$

Hence, it's unnecessary to further investigate the remaining paths on the list. Each of these paths, characterized by the prefix $\mathbf{u}_0^s[l]$, has a metric higher than that of $\mathbf{u}_0^s[l_{min}]$. Consequently, they cannot yield a codeword with a metric equal to or lower than the codeword in $\mathcal{C}_N(\mathbf{u}_0^s[l_{min}])$. Thus, retaining solely the path with the lowest metric at the s^{th} decoding step does not impact the performance of the list decoder. The next step consists in finding the best candidate $\mathbf{x}_{min}$.

To ensure the identification of the codeword $\mathbf{x}_{min}$ with metric $m_s(\mathbf{u}_0^s[l_{min}], \mathbf{y})$, it's essential that for all $j \in [s+1, N-1]$, $m_j(\mathbf{u}_0^j[l_{min}], \mathbf{y}) = m_s(\mathbf{u}_0^s[l_{min}], \mathbf{y})$. To achieve this, the chosen path at the $(s+1)^{th}$ decoding step must not increase its associated metric compared to the s^{th} decoding step. As demonstrated in Chapter 1, the path metric $m_{i+1}(\mathbf{u}_0^{i+1}[l])$ can be represented as follows:

$$m_{i+1}(\mathbf{u}_0^{i+1}[l]) = \begin{cases} m_i(\mathbf{u}_0^i[l]) & \text{if } \text{sign}(1-2\hat{u}_i(l)) = \text{sign}(L_j^{(n)}) \\ m_i(\mathbf{u}_0^i[l]) + |L_i| & \text{otherwise} \end{cases}$$

Thus, to ensure $m_j(\mathbf{u}_0^j[l_{min}], \mathbf{y}) = m_s(\mathbf{u}_0^s[l_{min}], \mathbf{y}) \forall j \in [s+1, N-1]$, it must hold that $\text{sign}((1-2\hat{u}_j[l])) = \text{sign}(L_j)$ for all $j \in [s+1, N-1]$. It means that employing a Successive Cancellation (SC) decoder on the path $\mathbf{u}_0^s$ for the final $N-s-1$ decoding steps ensures the discovery of $\mathbf{x}_{min}$ without compromising performance compared to employing Successive Cancellation List (SCL) decoding for the last $N-s-1$ steps.
From these observations, we propose a modification to the SCL decoding algorithm, named the Tailored Successive Cancellation List (T-SCL) decoder. This modification maintains performance while decreasing the computational complexity of the decoding process. Algorithm 2 provides a detailed overview of the proposed approach, which entails the following key steps:

- An SCL decoding process is performed and the L potential best paths are kept in the list for the first s enumeration steps.
- At the s^{th} enumeration step, the path with the least metric $\mathbf{u}_0^s[l_{min}]$ is selected. All the other paths remaining in the list are discarded.
- An SC decoding is performed on $\mathbf{u}_0^s[l_{min}]$ for the rest of the $N-s-1$ decoding steps and the final codeword is obtained.

Note that the proposed decoding process can be extended to Polarization-Aided Concatenated (PAC) codes and polar codes with Dynamic Frozen Bit (DFB) schemes. The steps remain identical, with the only variation being the consideration of precoding for $\mathbf{u}_0^i$ at each decoding step. The fundamental approach of selecting the path with the least metric after the last frozen bit and performing SC decoding on the chosen path remains consistent

The reduction in terms of computational complexity arises from exploring only one path after the last frozen bit, as opposed to the exploration of L paths in a conventional SCL decoder. This eliminates the need to explore $L-1$ additional paths and avoids the

Algorithm 2: T-SCL decoding process

Input: Polar code $\mathcal{C}(N, K, \mathcal{F}), L$

$s \leftarrow \max(\mathcal{F})$;
for $i \in [\![0, s]\!]$ **do**
  Perform step i of SCL
end
$\mathbf{u}_0^s[l_{min}] \leftarrow$ path in the list with least metric
for $i \in [\![s+1; N-1]\!]$ **do**
  Perform SC decoding of u_i on path $\mathbf{u}_0^i[l_{min}]$
end
return $\mathbf{u}_0^{N-1}[l_{min}]$

N	128			256			512		
K	32	64	96	64	128	192	128	256	384
s	113	98	73	229	201	193	481	417	385

TABLE 4.1: Values of s for 5G rate-profiling

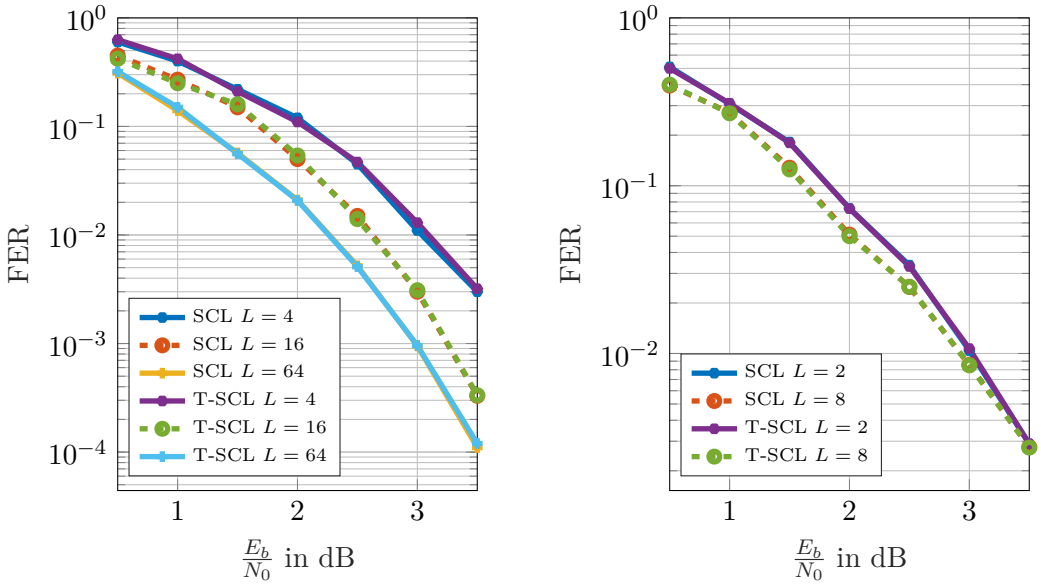
sorting operation of metrics to retain paths with the lowest metric in the list. The extent of computational complexity reduction varies depending on the position of the last frozen bit s. Smaller values of s result in fewer SCL decoding steps and greater reductions in terms of computational complexity.

The level of reduction differs based on the specific code construction and primarily relies on the value of s. Generally, in conventional rate-profiled polar codes, smaller values of s correspond to higher code rates. Tables 4.1 and 4.2 illustrates the values of s for different code lengths and rates in the case of 5G and RM rate-profilings. Notably, lower code rates tend to have higher values of s. Additionally, it's observed that s is greater for RM rate-profiles compared to 5G rate-profiles. For instance, for a (128,64) polar code, $s = 98$ for 5G rate-profiling compared to $s = 113$ for RM rate-profiling.

Figures 4.7a and 4.7b, show the evolution of the FER of polar codes under 5G and RM rate-profilings for a $(128, 64)$ code under different list sizes. Figures show that the FER curves superimpose in all the different cases which further confirms the proposed approach. The same results are observed for larger code sizes.

N	128			256			512		
K	29	64	99	37	93	163	130	256	382
s	121	113	97	249	241	225	497	481	449

TABLE 4.2: Values of s for RM rate-profiling



(a) 5G construction (b) RM construction

Figure 4.7: Performance of (128,64) Polar code under T-SCL decoding

4.3.2.2 Tailoring in the case of CA-SCL algorithm

In this section, tailoring within the context of the CA-SCL algorithm is considered. Initially, it is shown why the strategy outlined for SCL decoding cannot directly be applicable to CA-SCL decoding. Then, the adapted approach tailored specifically for CA-SCL decoding is explained.
The difference between SCL and CA-SCL decoding resides in the fact that the path with the least metric is not necessarily the one chosen at the end of the decoding. Indeed, this path also has to check the CRC. Subsequently, choosing the path with the least metric after the last frozen bit and only exploring that specific path does not apply. Indeed, if this path doesn't check the CRC, the decoding process is declared to be failed. This implies that the paths with the best metrics have to be kept in the list until the end of the decoding.

Therefore, we propose a simplification of the CA-SCL decoding algorithm in which the "tail" of the sequence is decoded with SC decoding. More precisely, starting from a certain index $\omega = N - T$, the path metrics are no longer updated nor is the metric sorting process. Instead, the L paths are kept and an SC decoding is performed independently on each one of these paths. We also introduce S as the set of the indices where splitting is considered on the tail. In this case, $|S|$ defines the number of splits that are performed on the tail. Algorithm 1 gives the details of the T-CA-SCL decoding algorithm. The

decoding process is performed into two major steps:

1. The first step consists in decoding u_0^{N-T}. During this decoding step, T-CA-SCL algorithm is identical to an SCL decoder. We denote $\mathcal{L}$ the list of the candidate paths at the end of this step. This phase is illustrated for $L = 4$ in Figure 4.8.

2. The second step starts with the decoding of the $N - T + 1^{th}$ bit. During this step, two different processes are considered depending on the decoding step:

   - $i \notin S$: typically, the bits at the end of the sequence lead to one path being much more favorable than the other. This means that in most cases, it is very unlikely to make a false decision by choosing the favorable path. At this point, an SC decoder is performed on each one of the previously kept paths separately. Thus, no path sorting nor metric updating are needed.
   - $i \in S$: sometimes, the bits at the end of sequence can be unreliable. Hard decision making at this point may lead to non negligible errors. In this particular case, an SCL decoding is performed on those specific bits. As an example, figure 4.8 shows a splitting on the tail for u_{S_1}.

Note that frozen bits are set to zero and are omitted in Algorithm 1. A CRC check is then applied to return the decoded sequence. By this way, the T-CA-SCL algorithm can achieve lower computational complexity by removing both metrics sorting and updating for a significant number of decoding steps.

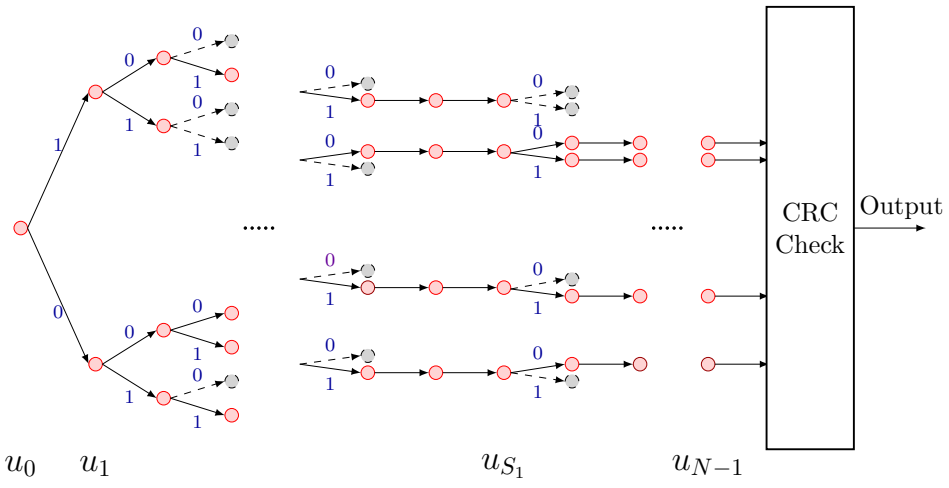


Figure 4.8: T-CA-SCL decoding process illustration.

Algorithm 3 implies that, for each investigated polar code, the parameters T and S have to be fixed. Since in contrast to the SCL decoding where the tailoring start can

Algorithm 3: T-CA-SCL decoding

Input: Polar code $\mathcal{P}(N, K, \mathcal{F}), T, S, L$

```
for i ∈ ⟦0; N − T⟧ do
    Perform step i of SCL
end
L ← list of the SCL paths
for i ∈ ⟦N − T + 1; N⟧ do
    if i ∉ F and i ∉ S then
        foreach l ∈ L do
            SC decoding of u_i on path l
        end
    else
        Perform step i of SCL
    end
end
valid ← The path with the least metric that verifies the CRC
if valid = false then
    Report a decoding failure
return u[valid]
```

be determined analytically, in the case of a CA-SCL decoding, there is no direct way to obtain T. Hence, the parameters T and S are determined thanks to a heuristic approach. The heuristic described in Algorithm 4 performs Monte Carlo simulation on 100000 frames with a T-CA-SCL decoder and gradually increases the tail size T until the FER performance decreases. As some unreliable channels are located at the end of the sequence, in order to further increase T without significantly degrading the performance, the decoder can perform list decoding on the unreliable bits with indices S.

Algorithm 3 was applied to 6 different polar codes ($R = \{1/2, 3/4\}$ and $N = \{128, 256, 512\}$) selected from the 5G standard [9]. Table 4.3 summarizes T and S values obtained for these codes. The ratio $\rho = (T - |S|)/K$ represents the proportion of bits decoded without splitting. Depending on the considered code, this ratio can reach as much as ∼45%. In other words, almost half of the information bits are decoded without metric update nor metric sorting without degrading the decoding performance.

Error rate Monte Carlo simulations were performed on the six selected polar codes. The decoding performance curves in Figures 4.9a and 4.9b confirm that the T and S parameter selection enables the T-CA-SCL algorithm to perform very close to CA-SCL algorithm. The same trends were observed for larger list sizes.

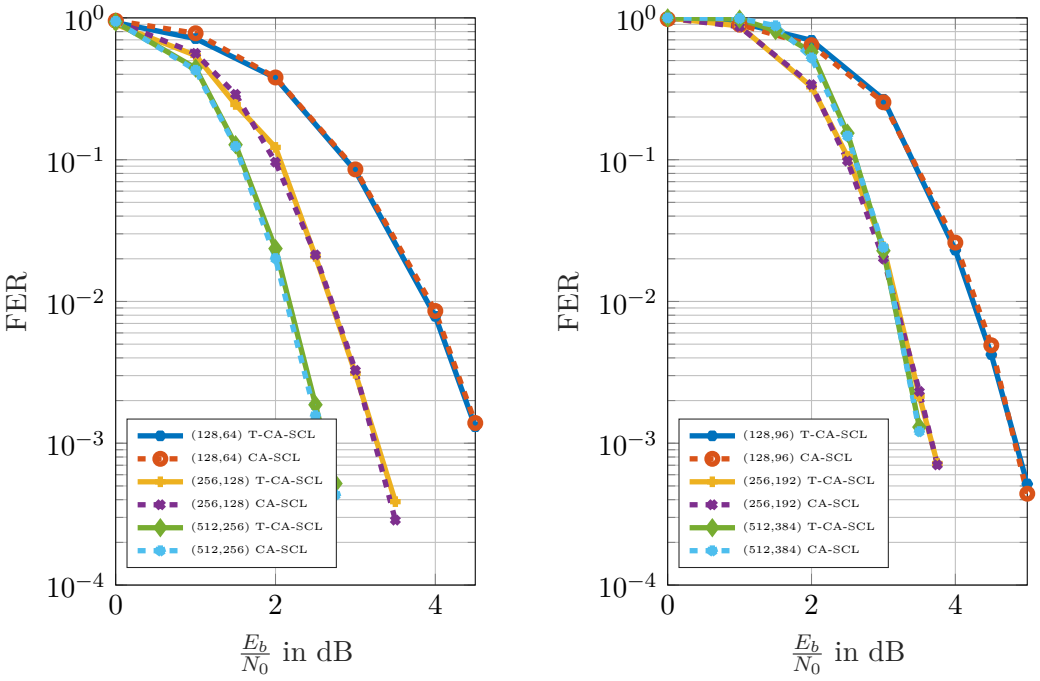
Algorithm 4: T-CA-SCL algorithm parameter generation

Input: Polar code $\mathcal{P}(N, K, \mathcal{F})$

$S = \{\oslash\}$, $T = 0$, $j = 0$;
Y : set of 10000 noisy codewords;
FER $\leftarrow$ T-CA-SCL($\mathcal{P}$, T, S, **Y**);
while $FER \sim FER_{CA-SCL}$ **do**
    FER $\leftarrow$ T-CA-SCL($\mathcal{P}$, T, S, **Y**);
    if $FER \sim FER_{CA-SCL}$ **then**
        $T \leftarrow T + 1$;
    else
        $s_j = T$;
        $j \leftarrow j + 1$;
    end
end
return (T, S)

N	128	256	512	128	256	512
K	64	128	256	96	192	384
R	1/2	1/2	1/2	3/4	3/4	3/4
S	113	209,225	289	97	$\oslash$	$\oslash$
$\lvert S \rvert$	1	2	1	1	0	0
T	23	54	93	45	87	126
$\rho(\%)$	34	36	36	46	45	32

TABLE 4.3: Polar code and T-CA-SCL decoding parameters



(a) $R = 1/2$ (b) $R = 3/4$

Figure 4.9: Polar codes decoding: CA-SCL versus T-CA-SCL for $L = 8$

4.4 About the average list size that reaches ML performance

In the previous section, it was demonstrated that transitioning from SCL decoding to SC decoding starting from a specific index maintains the same performance level as a list decoder. Consequently, conducting a simple SC decoding on particular positions ensures the overall performance of the SCL decoder. Leveraging these insights, this section aims to estimate the average list size required during each decoding step for a particular code construction to achieve ML decoding performance. The authors of [45] presented innovative entropy-based methods for estimating lower and upper bounds on an information-theoretical quantity that approximates the average list size required by the SCL decoder to achieve ML performance. Although these bounds are easily computed using density evolution, the proposed upper bound does not consider the impact of frozen bits on the list size evolution, resulting in reduced accuracy. Additionally, the proposed lower bound underestimates average list sizes for certain configurations. Another limitation of [45] is the exclusion of the effects of precoding.

In this section, we introduce a new upper bound on the average list size required by the

SCL decoder to achieve ML performance. This method analyzes the properties of the prefixes remaining in the list at each decoding step i. It also accounts for the effects of pre-transformation and can be generalized to rate-compatible pure and pre-transformed polar codes. A significant advantage of the proposed method is that it simultaneously calculates both the distance properties and the average list size needed to achieve ML performance.

4.4.1 Difference to True Path Metric (DTPM)

In order to highlight the effect of the list size during SCL decoding, we first start by decomposing the error probability of an SCL decoder into two terms. Then, we focus on the term related to the list size. Let us denote by C_{ML} the event of correct decoding using a ML decoder and by E_{SCL} the event of failed decoding using an SCL algorithm. Then the probability of failed decoding using an SCL decoder can be expressed as:

$$\begin{aligned}\mathbb{P}(E_{SCL}) &= \mathbb{P}(E_{SCL}|C_{ML})\mathbb{P}(C_{ML}) + \underbrace{\mathbb{P}(E_{SCL}|\bar{C}_{ML})}_{=1}\mathbb{P}(\bar{C}_{ML}) \\ &= \mathbb{P}(E_{SCL}|C_{ML})\mathbb{P}(C_{ML}) + \mathbb{P}(\bar{C}_{ML})\end{aligned} \tag{4.3}$$

Equation (4.3) yields that the probability of error of SCL decoding is expressed as the sum of two terms:

- The term $\mathbb{P}(\bar{C}_{ML})$ translates the probability of ML decoding failure and is therefore independent of the list size of the SCL decoder.
- The term $\mathbb{P}(E_{SCL}|C_{ML})\mathbb{P}(C_{ML})$ represents the probability that the SCL decoder fails while ML decoding would have succeeded. This scenario typically occurs when the true codeword is no longer present in the list at the end of the decoding. It means that the correct decoding path was excluded from the list at a specific decoding step i. Excluding the correct decoding path from the list at a given decoding step precludes its re-inclusion, leading to the decoding error.

In the following, we describe the mechanism leading to the ejection of the correct decoding path from the list at a particular decoding step. It is the starting point leading to the evaluation of the average list size that is necessary at each decoding step to achieve ML performance i.e. so that $\mathbb{P}(E_{SCL}|C_{ML}) \approx 0$. In all the following, and for simplification matters, we will suppose that the true transmitted codeword is the all-zero codeword. This hypothesis remains valid given the linearity of the code. Let us consider the i^{th} decoding step of an SCL decoder with a list size L. The correct decoding path $\mathbf{0}_0^i$ is

excluded from the list if there exists more than $L-1$ paths $\mathbf{u}_0^i[l]$ that have a path metric less than the metric of the correct decoding path. In other terms, given the list of paths $\mathcal{L}$:

$$\mathbf{0}_0^i \notin \mathcal{L} \Leftrightarrow \exists \mathcal{M} \subseteq \mathcal{L} | \quad |\mathcal{M}| > L-1 \quad \text{and} \quad m_i(\mathbf{u}_0^i[l]) < m_i(\mathbf{0}_0^i) \quad \forall \mathbf{u}_0^i[l] \in \mathcal{M} \tag{4.4}$$

Given a path $\mathbf{u}_0^i[l]$ at a decoding step i, the aim is to evaluate the probability that its associated path metric is lower than the path metric of the true path. We introduce the notion of Difference to True Path Metric (DTPM) of a path $\mathbf{u}_0^i[l]$ defined as:

$$\psi_i(\mathbf{u}_0^i[l]) = m_i(\mathbf{u}_0^i[l]) - m_i(\mathbf{0}_0^i) \tag{4.5}$$

Using Equation (4.2), $\psi_i(\mathbf{u}_0^i[l])$ can be written as:

$$\psi_i(\mathbf{u}_0^i[l]) = \frac{1}{2\sigma^2} \min_{\mathbf{x} \in \mathcal{C}_N(\mathbf{u}_0^i[l])} \left(||\mathbf{y}-\mathbf{x}||_2^2\right) - \frac{1}{2\sigma^2} \min_{\mathbf{x}' \in \mathcal{C}_N(\mathbf{0}_0^i)} \left(||\mathbf{y}-\mathbf{x}'||_2^2\right) \tag{4.6}$$

In all the following we make the following hypothesis

Hypothesis 4.4.1.

1. $\forall i \in [\![0, N-1]\!]$, $\underset{\mathbf{x} \in \mathcal{C}_N(\boldsymbol{0}_0^i)}{\operatorname{argmin}} \left(||\mathbf{y}-\mathbf{x}||_2^2\right) = \boldsymbol{0}_0^{N-1}$ *This means that:*

$$\forall i \in [\![0, N-1]\!], \min_{\mathbf{x} \in \mathcal{C}_N(\boldsymbol{0}_0^i)} \left(||\mathbf{y}-\mathbf{x}||_2^2\right) \approx ||\mathbf{y}-\boldsymbol{1}||_2^2$$

In other words, for each decoding step, we suppose that the closest codeword to the received codeword is the all-zero codeword. Note that this is not always true since at early decoding steps, there can be other codewords in the coset with a closer Euclidean distance to the received codeword. This means that in general,

$$\min_{\mathbf{x} \in \mathcal{C}_N(\boldsymbol{0}_0^i)} \left(||\mathbf{y}-\mathbf{x}||_2^2\right) \leq ||\mathbf{y}-1||_2^2$$

This hypothesis therefore tends to underestimate $\psi_i(\mathbf{u}_0^i[l])$.

2. $\forall i \in [\![0, N-1]\!]$, $\underset{\mathbf{x} \in \mathcal{C}_N(\mathbf{u}_0^i)}{\operatorname{argmin}} \left(||\mathbf{y}-\mathbf{x}||_2^2\right) = \mathbf{x}^* = 1-2\mathbf{c}^*$ *such that* $w(c^*) = w^*(\mathcal{C}_N(\mathbf{u}_0^i))$.
This means that:

$$\forall i \in [\![0, N-1]\!], \min_{\mathbf{x} \in \mathcal{C}_N(\mathbf{u}_0^i)} \left(||\mathbf{y}-\mathbf{x}||_2^2\right) \approx \min_{\mathbf{x} \in \mathcal{X}^*} \left(||\mathbf{y}-\mathbf{x}||_2^2\right)$$

where $\mathcal{X}^*$ is the set defined by:

$$\mathcal{X}^* = \left\{\mathbf{x} = 1 - 2\mathbf{c} | \mathbf{c} \in \mathcal{C}_N(\mathbf{u}_0^i) \quad and \quad w(\mathbf{c}) = w^*(\mathcal{C}_N(\mathbf{u}_0^i))\right\}$$

Stated differently, it is assumed that during each decoding step, the nearest codeword of a coset $\mathcal{C}_N(\mathbf{u}_0^i)$ to the received codeword is one of the codewords with the minimum weight within that coset. This assumption is justified by the observation that codewords with the minimum weight within a particular coset are more likely to have a smaller Euclidean distance compared to other codewords with higher weights within the same coset.

Based on these hypothesis, $\psi_i(\mathbf{u}_0^i[l])$ can be expressed as:

$$\begin{aligned}\psi_i(\mathbf{u}_0^i[l]) &\approx \frac{1}{2\sigma^2} \min_{\mathbf{x}\in\mathcal{X}^*} \left(||\mathbf{y}||_2^2 + ||\mathbf{x}||_2^2 - 2\sum_{j=0}^{N-1} y_j x_j \right) - ||\mathbf{y}||_2^2 - ||\mathbf{1}||_2^2 + 2\sum_{j=0}^{N-1} y_j \\ &\approx \frac{1}{\sigma^2} \min_{\mathbf{x}\in\mathcal{X}^*} \sum_{j=0}^{N-1} y_j(1 - x_j)\end{aligned} \tag{4.7}$$

Given that $x_j = 1 - 2c_j$, $\psi_i(\mathbf{u}_0^i[l])$ is finally expressed as:

$$\psi_i(\mathbf{u}_0^i[l]) = \min_{\mathbf{c}\in\mathcal{C}^*} \sum_{j=0}^{N-1} \frac{2}{\sigma^2} y_j c_j \tag{4.8}$$

where $\mathcal{C}^*$ defines the set:

$$\mathcal{C}^* = \left\{\mathbf{c} \in \mathcal{C}_N(\mathbf{u}_0^i[l]) | w(\mathbf{c}) = w^*(\mathcal{C}_N^i(\mathbf{u}_0^i[l]))\right\}$$

The objective is to evaluate the probability $\mathbb{P}(\psi_i(\mathbf{u}_0^i[l]) \leq 0)$. Note that since we underestimate $\psi_i(\mathbf{u}_0^i[l])$ when using Hypothesis 4.4.1, the probability $\mathbb{P}(\psi_i(\mathbf{u}_0^i[l]) \leq 0)$ is overestimated. Hence, we provide an upper bound on $\mathbb{P}(\psi_i(\mathbf{u}_0^i[l]) \leq 0)$.
Let us first consider the term $z = \frac{2}{\sigma^2}\sum_{j=0}^{N-1} y_j c_j$. Since we made the hypothesis that $\forall \mathbf{c} \in \mathcal{C}^*, w(\mathbf{c}) = w^*(\mathcal{C}_N^i(\mathbf{u}_0^i[l])) = w_{i,l}^*$, z is the sum of $w_{i,l}^*$ random variables $\sim \mathcal{N}(1, \sigma^2)$. Therefore, z follows a Gaussian law with mean $\frac{2w_{i,l}^*}{\sigma^2}$ and variance $\frac{4w_{i,l}^*}{\sigma^2}$, i.e. z with distribution $\mathcal{N}(\mu_{i,l} = \frac{2w_{i,l}^*}{\sigma^2}, \sigma_{i,l} = \frac{4w_{i,l}^*}{\sigma^2})$.

Authors of [89] detailed the expression of the Probability Density Function (PDF) of the minimum of $A_{i,l}$ Gaussian random variables with identical means and variances. In particular, when the random variables are supposed to be correlated with an identical

correlation factor $\rho_{i,l}$, the CDF of $\psi_i(\mathbf{u}_0^i[l])$ and the PDF of $\psi_i(\mathbf{u}_0^i[l])$ are respectively expressed as:

$$f_{\psi_i(\mathbf{u}_0^i[l])}(v) = \int_{-\infty}^{+\infty} \frac{1}{\sqrt{\rho_{i,l}}\sigma_{i,l}} \tilde{f}_{\psi_i(\mathbf{u}_0^i[l])}(t)\, \phi\left(\frac{\sqrt{1-\rho_{i,l}}t - \frac{v-\mu_{i,l}}{\sigma_{i,l}}}{\sqrt{\rho_{i,l}}}\right) dt \tag{4.9}$$

$$F_{\psi_i(\mathbf{u}_0^i[l])}(v) = \int_{-\infty}^{+\infty} \frac{\sqrt{1-\rho_{i,l}}}{\sqrt{\rho_{i,l}}\sigma_{i,l}} \tilde{F}_{\psi_i(\mathbf{u}_0^i[l])}(t)\, \phi\left(\frac{\sqrt{1-\rho_{i,l}}t - \frac{v-\mu_{i,l}}{\sigma_{i,l}}}{\sqrt{\rho_{i,l}}}\right) dt \tag{4.10}$$

where $\Phi(v)$ and $\phi(z)$ denote the CDF and the PDF of the law $\mathcal{N}(0,1)$ respectively. Consequently, $\tilde{F}_{\psi_i(\mathbf{u}_0^i[l])}(v)$ and $\tilde{f}_{\psi_i(\mathbf{u}_0^i[l])}(v)$ denote the CDF and the PDF the law of the minimum of A_i independent normal variables ($\rho_{i,l} = 0$) :

$$\tilde{F}_{\psi_i(\mathbf{u}_0^i[l])}(v) = 1 - (1 - \Phi(v))^{A_{i,l}-1} \tag{4.11}$$

$$\tilde{f}_{\psi_i(\mathbf{u}_0^i[l])}(v) = A_{i,l}(1 - \Phi(v))^{A_{i,l}-1}\phi(v) \tag{4.12}$$

The computation of A_i is easily obtained from the results of Chapter 2. Indeed, they enable to compute, for a specific polar coset its minimum weight and the number of codewords with minimum weight. The main difficulty in Equation (4.9) is the computation of the correlation $\rho_{i,l}$. This computation is presented and justified in the next section.

4.4.2 Correlation of codewords with minimum weight of a coset

In this section, we focus on the computation of the correlation factor ρ_i between the elements of minimum weight of a specific coset. The problematic is the following: given a number A of random variables $z_i = \sum_{j=0}^{N-1} \frac{2}{\sigma^2} y_j c_{ij}$ such that $w(\mathbf{c}_i) = d$, we want to express $\bar{\rho}$ the average correlation between the random variables $z_i, i \in [\![0, A-1]\!]$, i.e.:

$$\bar{\rho} = \frac{\sum_{i=0}^{A-1} \sum_{\substack{k=0 \\ k\neq i}}^{A-1} corr(z_i, z_k)}{A(A-1)} \tag{4.13}$$

The choice of computing the average value of the correlation comes from the fact that to the best of the author's knowledge, there is no direct way to characterize individually the correlation of each couple of elements in a coset unless knowing the exact codewords, leading to high computational complexity. We propose in the following an approach that can be applied for this computation. First, let us express the correlation coefficient of two random variables z_i and z_k:

$$\begin{aligned}
\mathrm{corr}(z_i, z_k) &= \frac{\mathrm{cov}(z_i, z_k)}{\sqrt{\mathbb{V}(z_i)\mathbb{V}(z_k)}} \\
&= \frac{4}{\sigma^4} \frac{\sum_{j=0}^{N-1} \sum_{l=0}^{N-1} c_{ij} c_{kl} \mathrm{cov}(y_j, y_l)}{\sqrt{\mathbb{V}(z_i)\mathbb{V}(z_k)}} \\
&= \frac{4}{\sigma^4} \frac{\sum_{j=0}^{N-1} c_{ij} c_{kj} \mathbb{V}(y_j)}{\sqrt{\mathbb{V}(z_i)\mathbb{V}(z_k)}} \\
&= \frac{\sum_{j=0}^{N-1} c_{ij} c_{kj}}{d}
\end{aligned} \tag{4.14}$$

Equation (4.14) translates that the correlation of z_i and z_k essentially depends the number of positions for which $c_{ij} = c_{kj} = 1$, i.e. on the number of common 'ones' of the two codewords $\mathbf{c}_i$ and $\mathbf{c}_k$. $\bar{\rho}$ can be further expressed as:

$$\begin{aligned}
\bar{\rho} &= \frac{\sum_{i=0}^{A-1} \sum_{\substack{k=0 \\ k \neq i}}^{A-1} \sum_{j=0}^{N-1} c_{ij} c_{kj}}{dA(A-1)} \\
&= \frac{\sum_{j=0}^{N-1} \sum_{i=0}^{A-1} \sum_{\substack{k=0 \\ k \neq i}}^{A-1} c_{ij} c_{kj}}{dA(A-1)} \\
&= \frac{\sum_{j=0}^{N-1} \left(\sum_{i=0}^{A-1} \sum_{\substack{k=0 \\ k \neq i}}^{A-1} c_{ij} c_{kj} \right)}{dA(A-1)} \\
&= \frac{\sum_{j=0}^{N-1} \alpha_i(\alpha_i - 1)}{dA(A-1)}
\end{aligned} \tag{4.15}$$

where α_i is the number of codewords for which the i^{th} codeword bit is equal to 1. In the following, we propose a method that enables the computation of α_i.

4.4.3 Determination of α_i

In this section, we discuss the determination of the common number of ones the i^{th} codeword with minimum weight of a coset. This approach relies on the same message passing formalism detailed in Chapter 2 for the computation of the minimum weight and the associated number of codewords of polar cosets. Let us denote by $\mathcal{C}_N^{x_j=b}(\mathbf{u}_0^i)$ the subset of the polar coset $\mathcal{C}_N(\mathbf{u}_0^i)$ defined as:

$$\mathcal{C}_N^{x_j=b}(\mathbf{u}_0^i) = \left\{\mathbf{x} \in \mathcal{C}_N(\mathbf{u}_0^i) | x_j = b\right\} \tag{4.16}$$

where $b \in \{0, 1\}$. Starting with an example, we show how $\mathcal{C}_N^{x_j=0}(\mathbf{u}_0^i)$ and $\mathcal{C}_N^{x_j=1}(\mathbf{u}_0^i)$ can be computed for each $j \in [\![0, N-1]\!]$. We denote by $A_N^*(\mathcal{C}_N^{x_j=b}(\mathbf{u}_0^i))$ the MWEF of $\mathcal{C}_N^{x_j=b}(\mathbf{u}_0^i)$.

Example 4.4.2. *Figure 4.10 shows the MWEF factor graph of bit u_4 for a polar code with $N = 8$, $\mathbf{p} = [0, 0, 0, 0, 0, 1, 1, 1]$. The message passing enabling the computation of*

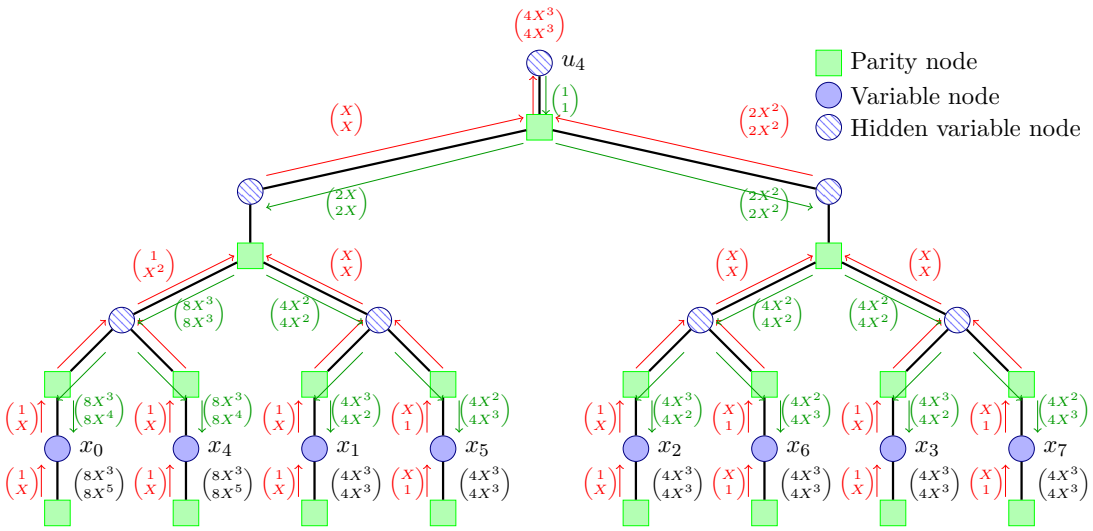


Figure 4.10: Tanner Graph of u_4 decoding for a polar code with $N = 8$

messages represented in red enables the computation of $A_8^*(\mathbf{u}_0^3, 0) = 4X^3$ and $A_8^*(\mathbf{u}_0^3, 1) = 4X^3$ in this case. The MWEFs $A_N^*(\mathbf{u}_0^i | x_j = 0)$ and $A_N^*(\mathbf{u}_0^i | x_j = 1)$ $\forall j \in [\![0, N-1]\!]$ are exactly defined by the messages arriving at the leaf nodes after the message passing represented in green. The final messages taking into account the effect of the different x_i are represented in black. For instance, in the case of x_0, $A_N^*(\mathbf{u}_0^3 | x_0 = 0) = 8X^3$ and

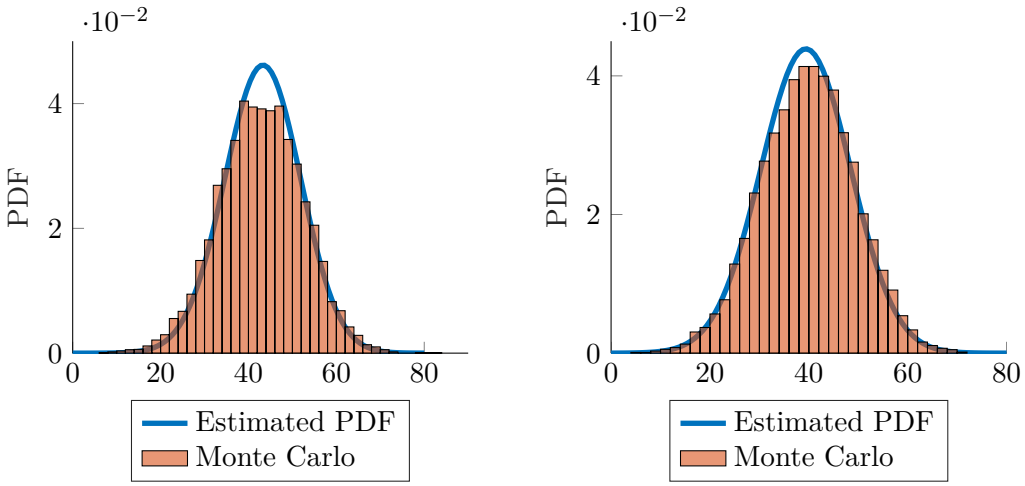
$A_N^*(\mathbf{u}_0^3|x_0=1)=8X^5$. *This means that all* 8 *codewords with a minimum weight equal to* 3 *have* $x_0=0$ *since the codewords with* $x_0=1$ *have a minimum weight equal to* 5. *In the case of* x_1, $A_N^*(\mathbf{u}_0^3|x_1=0)=4X^3$ *and* $A_N^*(\mathbf{u}_0^3|x_1=1)=4X^3$. *This means that there are* 4 *codewords of the coset* $\mathcal{C}_N(\mathbf{u}_0^3)$ *for which* $x_1=0$ *and* 4 *for which* $x_1=1$. *In this case, knowing that there are* 4 *codewords of the coset with* $x_1=1$, *the number of codeword couples having both* $x_1=1$ *is equal to* $\frac{4\times3}{2}=6$. *This computation of the different* $\alpha_j \quad \forall j\in[\![0,N-1]\!]$ *concludes the computation of the correlation factor* ρ *in Equation* (4.15).

4.4.4 DTPM estimation results

In this section, estimations of $\psi_i(\mathbf{u}_0^i[l])$ are presented for different decoding steps and paths. We also show the impact of the minimum weight and associated number of occurrences. Figure 4.11 gives histograms of $\psi_i(\mathbf{u}_0^{72}[l_1])$ and $\psi_i(\mathbf{u}_0^{72}[l_2])$ as well as their associated PDFs using Equation (4.9) for a (128, 64) PAC code and $E_b/N_0=3dB$. $\psi_i(\mathbf{u}_0^{72}[l_1])$ and $\psi_i(\mathbf{u}_0^{72}[l_2])$ denote the DTPM estimations for two different paths with prefixes of 73 bits. The minimum weight, associated number of occurrences as well as the values of the correlation ρ are summarised in Table 4.4. The accuracy of the approach is justified in Figure 4.11 since the DTPM is well estimated in both of cases. Table 4.4 shows that both cosets have the same minimum weight and correlation factor. $\mathcal{C}_{128}(\mathbf{u}_0^{72}[l_2])$ has a higher number of elements with minimum weight than $\mathcal{C}_{128}(\mathbf{u}_0^{72}[l_2])$. The estimation of both DTPMs is shown in Figure 4.11. it shows that the distribution $\psi_i(\mathbf{u}_0^{72}[l_2])$ stochastically dominates the distribution $\psi_i(\mathbf{u}_0^{72}[l_1])$. This implies that the CDF $F_{\psi_i(\mathbf{u}_0^{72}[l_2])}(0)\geq F_{\psi_i(\mathbf{u}_0^{72}[l_1])}(0)$ resulting in a higher probability of keeping the path $\mathbf{u}_0^{72}[l_2]$ in the list. This can be explained by exploring the expressions of $\psi_i(\mathbf{u}_0^i[l])$. In fact, since $\psi_i(\mathbf{u}_0^i[l])$ is the random variable that is defined as the minimum of A random variables with the same PDF, increasing A results in the reduction of $F_{\psi_i(\mathbf{u}_0^i[l])}(0)$.

Coset	$\mathcal{C}_{128}(\mathbf{u}_0^{72}[l_1])$	$\mathcal{C}_{128}(\mathbf{u}_0^{72}[l_2])$
w^*	16	16
A^*	128	2048
ρ	0.49	0.49

TABLE 4.4: Distance properties of $\mathcal{C}_{128}(\mathbf{u}_0^{72}[l_1])$ and $\mathcal{C}_{128}(\mathbf{u}_0^{72}[l_2])$



(a) Case $A^* = 128$ (b) Case $A^* = 2048$

Figure 4.11: Histograms and PDFs of $\psi_{72}(\mathbf{u}_0^{72}[l_1])$ and $\psi_{72}(\mathbf{u}_0^{72}[l_2])$

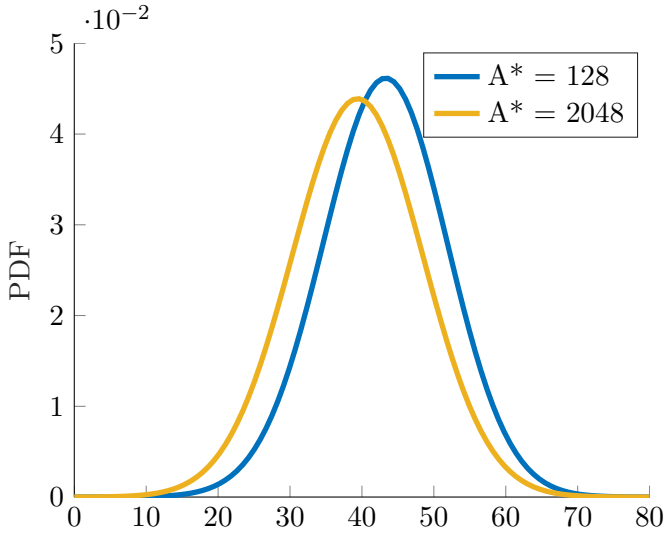


Figure 4.12: PDFs of both DTPMs

4.4.5 Average List size for ML decoding

In this section, we detail how the previous computations are used to evaluate an upper bound on the overall average list size capable of achieving the ML performance of polar codes. When considering a decoding step i, the minimum number of prefixes $\mathbf{u}_0^i$ that has to stay of the list in order to achieve ML performance is the number of prefixes having $\psi(\mathbf{u}_0^i) \leq 0$. In other terms, at this step, the list size L_i can be expressed as:

$$L_i = \sum_{\substack{\mathbf{u}_0^i \in \{0,1\}^i \\ u_j = 0, \forall \quad j \in \mathcal{F}}} \mathbb{1}(\psi(\mathbf{u}_0^i) \leq 0) \tag{4.17}$$

We denote by $\bar{L_i}$ the average list size that achieves the ML performance at the i^{th} decoding step. From Equation (4.17), $\bar{L_i}$ is expressed as:

$$\bar{L_i} = \mathbb{E}(L_i) = \sum_{\substack{\mathbf{u}_0^i \in \{0,1\}^i \\ u_j = 0, \forall \quad j \in \mathcal{F}}} \mathbb{P}(\psi_i(\mathbf{u}_0^i) \leq 0) \tag{4.18}$$

Note that since we computed an upper bound on $\mathbb{P}(\psi_i(\mathbf{u}_0^i) \leq 0)$, $\bar{L_i}$ is considered an upper bound on the average list size to reach ML performance.
Equation (4.18) implies that in order to evaluate each $\bar{L_i}$, the probability $\mathbb{P}(\psi_i(\mathbf{u}_0^i) \leq 0)$ has to be evaluated for a number $\gamma^i = |\left\{j \in \mathcal{I}^i\right\}|$ of prefixes. As γ^i can be very high, a simplification of $\bar{L_i}$ is proposed as follows:

$$\bar{L_i} \approx \sum_{\substack{\mathbf{u}_0^i \in \{0,1\}^i \\ u_j = 0, \forall j \quad \in \mathcal{F} \\ w^*(\mathcal{C}_N(\mathbf{u}_0^i)) \leq w_{end}}} \mathbb{P}(\psi_i(\mathbf{u}_0^i) \leq 0) \tag{4.19}$$

This simplification stems from the observation that the prefixes producing codewords with weights close to the code's minimum overall weight tend to represent the paths that were more probable to remain in the list. We designate w_{end} as a value marginally higher than the code's minimum weight.
Algorithm 5 details the different steps required in order to compute $\bar{L_i}$. The algorithm operates as follows:

1. During each step, the minimum weight and its associated number of occurrences are determined for each coset associated to the remaining prefixes in the list. The

cosets having a minimum weight greater than w_{end} are discarded.

2. For the remaining prefixes, the correlation factor ρ is calculated, and $\mathbb{P}(\psi_i(\mathbf{u}_0^i) \leq 0)$ is determined using Equations (4.10) and (4.11). It's worth noting that since $\mathbb{P}(\psi_i(\mathbf{u}_0^i) \leq 0)$ relies mainly on the minimum weight, the associated number of occurrences, and the correlation factor, it can be computed once for polar cosets sharing the same parameters.

3. The average list size is only computed until the last frozen bit. This is because, we showed in Section 4.3.2.1 that after the last frozen bit, if the true path persists in the list, it can be retrieved using solely an SC decoding process (i.e. list decoding with a list size equal to 1).

It's worth noting that Algorithm 5 shares significant similarities with previously proposed algorithms for computing the partial weight spectrum of polar codes, as both rely on the minimum weight and associated number of occurrences of polar cosets. Consequently, when determining the list size required for ML decoding, the partial weight spectrum can also be computed. It corresponds to a significant advantage, as the computations performed simultaneously serve for evaluating both the distance properties and the computational complexity of decoding process.

Algorithm 5: Computing the average list size required for ML decoding performance

Input: Polar code $\mathcal{C}(N, K, \mathcal{F}), w_{end}$

$s \leftarrow$ index of the last frozen bit
$L \leftarrow 1$
for $i \in [\![1; s]\!]$ **do**
  for $l \in [\![1; L]\!]$ **do**
    Compute w^* and A^* of the cosets in the list
    Discard the cosets for which $w^* > w_{end}$
    Compute the correlation factor ρ and $\mathbb{P}(\psi_i(\mathbf{u}_0^i) \leq 0)$
    $\bar{L}_i \leftarrow \sum_{\mathbf{u}_0^i \in \mathcal{L}} \mathbb{P}(\psi_i(\mathbf{u}_0^i) \leq 0)$
    $L \leftarrow |\mathcal{L}|$
  end
end
Return $\bar{L}$

4.4.6 Average list size estimation results

In this section, we compare the results obtained via Algorithm 5 to the ones obtained using Monte Carlo simulations. Figure 4.13 illustrates the evolution of the proposed

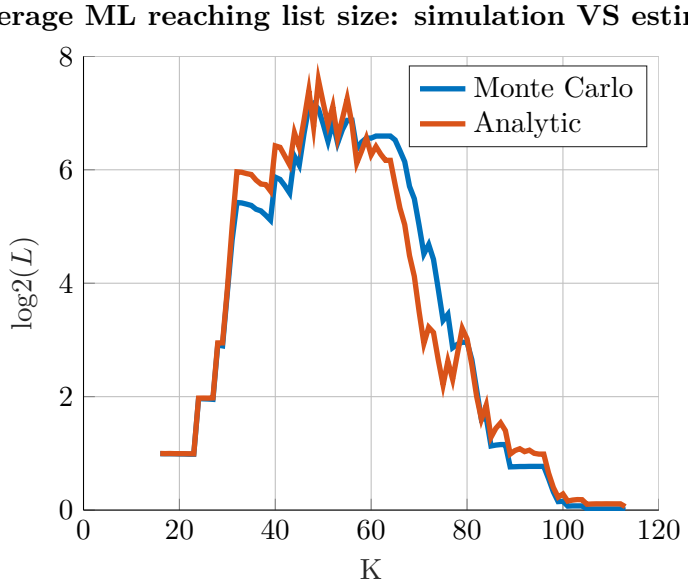


Figure 4.13: $\log_2(\bar{L}_i)$: Estimation VS simulation for $(128, 64)$ a PAC code under RM construction

upper bound for a $(128, 64)$ PAC code under RM construction with $\frac{E_b}{N_0} = 0.5dB$ and $w_{end} = 20$. We give on the same figure the evolution of $\log_2(\bar{L}_i)$ estimated using the described approach in Algorithm 5 as well as using a Monte-Carlo approach. Figure 4.13 shows that $\log_2(\bar{L}_i)$ estimated using Algorithm 5 is close to the estimated list size via Monte-Carlo simulations. Note that the estimation is represented for relatively low signal to noise ratio since when defining the required list size for reaching ML performance, we are more interested in the rare received codewords that need a large list size in order to be correctly recovered. Those rare events are more visible for low signal to noise ratios.

In order to understand why the RM configuration requires a larger list size in order to reach ML performance, we take a closer look on the DTPM at the first decoding steps. Figure 4.14 represents the pdf of the DTPM $\psi([\mathbf{0}_0^{14}1])$ for $\frac{E_b}{N_0} = 0.5$ (Note that in the case of RM construction, u_{15} is the first information bit). It is highlighted in the figure that $\psi([\mathbf{0}_0^{14}1])$ almost takes only negative values implying that $F_{\psi([\mathbf{0}_0^{14}1])}(0) \approx 1$. This means that the path $[\mathbf{0}_0^{14}1]$ is almost always kept in the list. The values of $f_{\psi([\mathbf{0}_0^{14}1])}(v)$ are controlled by $w^*(\mathcal{C}_{128}([\mathbf{0}_0^{14}1]))) = 16$ and $A^*(\mathcal{C}_{128}([\mathbf{0}_0^{14}1]))) = 281474976710656$. While the minimum weight of the coset is relatively high, the number of codewords of the coset with minimum weight is very high resulting in a higher probability of keeping the path in the list as explained in Section 4.4.4. In order to highlight the effect of precoding on the average list size for reaching ML performance, Figure 4.15 represents

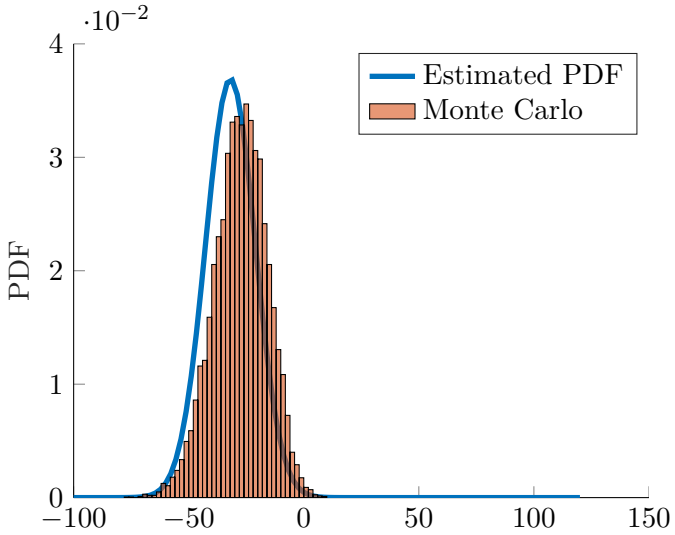


Figure 4.14: Representation of $f_{\psi([\mathbf{0}_0^{14}1])}(v)$

	A_{16}	A_{18}	A_{20}
Polar RM	94488	0	0
PAC RM	3120	2696	95828

TABLE 4.5: Number of low weight codewords in the $(128, 64)$ pure polar and PAC codes

the estimation of the average list size in the case of pure polar and PAC codes under RM constructions with $\frac{E_b}{N_0} = 0.5dB$ and $w_{end} = 20$. Figure shows that a pure polar under RM construction requires a smaller list size to achieve ML performance than a PAC codes. This validates the results presented in Chapter 1 (cf. Figures 1.11b and 1.13b) where a pure polar codes reaches its ML bound under SCL decoding with $L = 64$ while PAC code under RM construction reaches its ML bound under SCL decoding with $L = 256$. The number of low weight codewords of pure polar and PAC codes under RM construction are represented in Table 4.5. While PAC encoding considerably reduced the number of codewords with minimum distance 16, it leads to the formation of codewords with weights 18 and 20. Figure 4.16 shows the evolution of the pdf of $\psi(\mathbf{u}_0^{84}[l_1])$ and $\psi(\mathbf{u}_0^{84}[l_2])$ for a $(128, 64)$ PAC code under RM construction with parameters given in Table 4.6.

Coset	$\mathcal{C}_{128}(\mathbf{u}_0^{84}[l_1])$	$\mathcal{C}_{128}(\mathbf{u}_0^{84}[l_2])$
w^*	16	20
A^*	8	131072

TABLE 4.6: Distance properties of $\mathcal{C}_{128}(\mathbf{u}_0^{84}[l_1])$ and $\mathcal{C}_{128}(\mathbf{u}_0^{84}[l_2])$

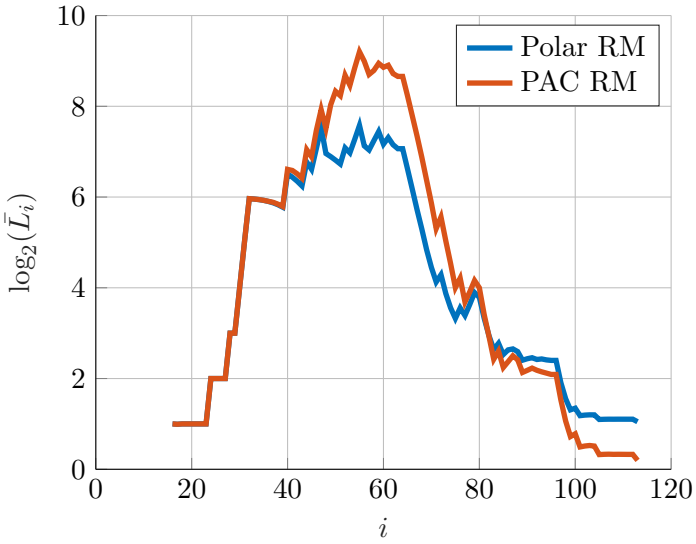


Figure 4.15: $\log_2(\bar{L}_i)$: Estimation for $(128, 64)$ pure polar and PAC codes under RM construction

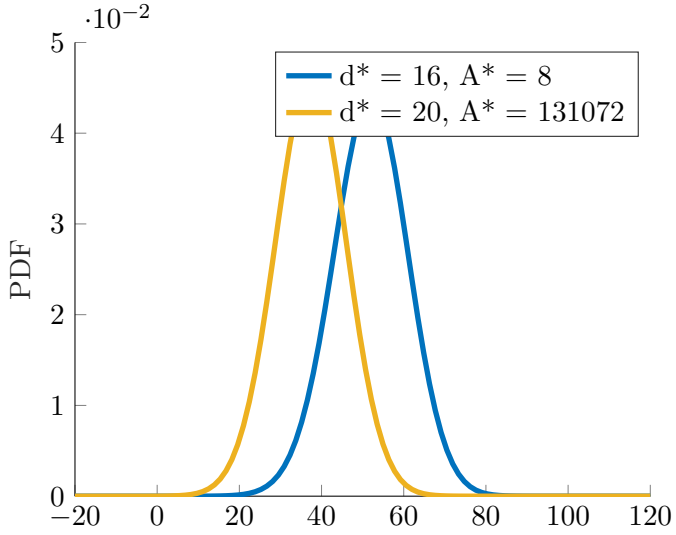


Figure 4.16: PDFs of $\psi(\mathbf{u}_0^{84}[l1])$ and $\psi(\mathbf{u}_0^{84}[l2])$

While $\mathcal{C}_{128}(\mathbf{u}_0^{84}[l1])$ has a lower minimum weight than $\mathcal{C}_{128}(\mathbf{u}_0^{84}[l2])$, the number of occurrences of codewords with minimum weight is much higher in $\mathcal{C}_{128}(\mathbf{u}_0^{84}[l1])$, making the distribution of $\psi(\mathbf{u}_0^{84}[l1])$ stochastically dominant on the distribution of $\psi(\mathbf{u}_0^{84}[l2])$.

In order to further highlight the effect of the rate-profiling on the list size required to reach the ML performance, we represent in Figure 4.17a the evolution of $\log_2(\bar{L}_i)$ for $(256, 128)$ PAC codes constructed using Algorithm 1 and a novel design aiming to reduce the average list size needed to reach ML performance based on freezing some of the

	A_{16}	A_{18}	A_{20}	A_{22}
PAC*	0	0	894	310
PAC proposed	6228	96	120224	1456
PAC+ [90]	13904	1824	307808	25728

TABLE 4.7: Number of low-weight codewords

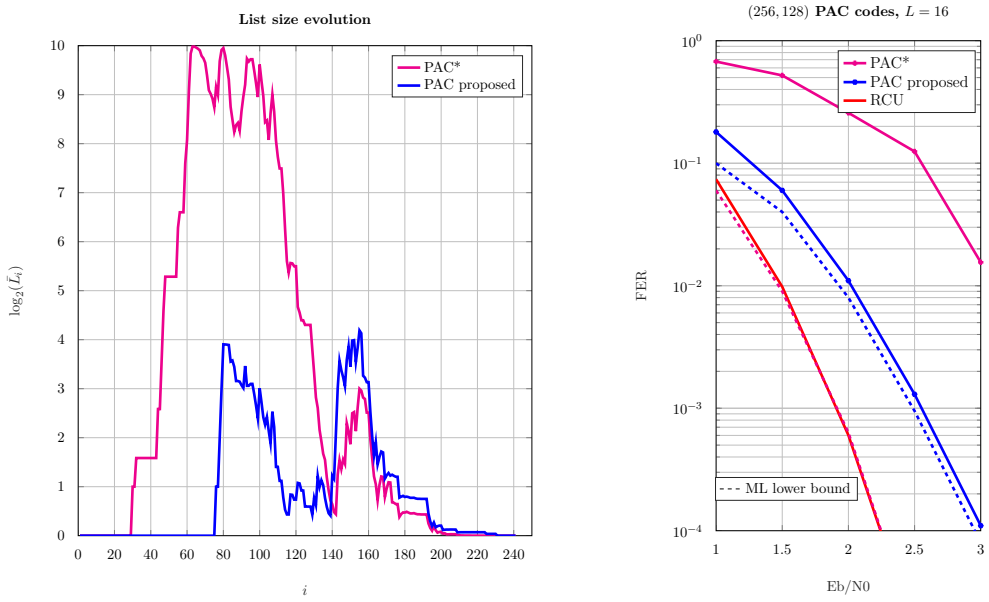
information bits that make the list size bigger and unfreezing some of the frozen bits. The new information set is

$$\mathcal{I}' = \{28, 31, 43, 45, 46, 47, 54, 58, 60, 86, 89, 90, 92, 106, 139\} \bigcup \mathcal{I}$$

$$\setminus \{120, 139, 141, 163, 172, 197, 202, 209, 210, 212, 216, 225, 226, 228, 232\}$$

and $E_b/N_0 = 1.5dB$. The partial weight spectrum of both configurations is represented in Table 4.7.

Figure 4.17a highlights the difference in $\bar{L}_i$ between both configurations. This modification on the other hand decreased the minimum distance from 20 to 16 as it can be observed in Table 4.7. The performance of the proposed PAC code is compared to the performance of the PAC+ [90] under list decoding of size $L = 32$ in Figure 4.17b. Figures shows that the proposed PAC construction outperforms the PAC+. In fact, it can be observed from Table 4.7 that $A^* = 6628$ for our construction, whereas it is equal to 13904 for the PAC+.



(a) List size evolution

(b) Performance of PAC codes

Figure 4.17: List size evolution and performance of $(256, 128)$ PAC codes under SCL decoding with $L = 16$

4.5 Conclusion

This chapter was dedicated to the definition of the various parameters necessary to estimate the list size requirement for achieving ML performance under SCL decoding.

Initially, we introduced an algorithm that facilitates code constructions that solely consider the distance properties of pre-transformed polar codes. This construction is based on putting into practice the results of the previous chapters that allow the determination of the distance properties of polar codes. While this construction showed promising performance for short block lengths under list decoding with moderate list sizes, it resulted in poor performance under SCL decoding with moderate list sizes for larger block lengths. This observation prompted the motivation to determine the different parameters that impact the average list size required to achieve ML performance for a given code construction.

Our first contribution consisted into determining when the list decoding becomes unnecessary. This approach relied on determining the indexes starting from which list decoding can be replaced by an SC decoder without altering the performance of polar codes. This approach was first presented for SCL decoder and then extended to CA-SCL decoder.

The next contribution involved estimating an upper bound on the required list size to achieve ML performance under SCL decoding. The process began by expressing the condition necessary to keep the true decoding path within the list at each decoding step, leading to an analytical expression of newly defined DTPMs for the different prefixes at each step of list decoding. By identifying the various parameters that influence these DTPMs, an estimation of an upper bound on the required list size was formulated. This bound accounts for the impact of pre-transformation and adjusts to the specifics of each code construction. Experimental results validate the accuracy of the proposed bound and elucidate the impact of the code construction on the list size. Additionally, a newly designed rate-profile is introduced to demonstrate the method's effectiveness for code design, taking into account both distance properties and SCL computational complexity.

Conclusions and perspectives

Conclusions

Since their discovery by Arikan in 2008, polar codes have garnered increasing attention owing to their attractive property of asymptotically approaching Shannon's capacity. Their simplicity in encoding and decoding, particularly when compared to other codes, along with their adaptability to puncturing and shortening techniques, further contribute to their appeal. However, achieving Shannon's capacity typically involves very large block lengths. In practice, it is challenging for the realm of short to moderate block lengths.

This thesis manuscript primarily addresses two scientific challenges. Firstly, it focuses on the efficient determination of distance properties of polar codes, considering various transformations that can be applied to them. Secondly, it delves into the computational complexity of decoding under a given code construction. The objective is to propose methods that can help to establish a trade-off between code performance under optimal decoding and the computational complexity of decoders approaching optimal decoding.

In the second chapter, the focus was on addressing the challenge of determining the minimum distance and its associated number of occurrences for both pure and pre-transformed polar codes. The chapter demonstrated the feasibility of computing the minimum weight and the corresponding number of occurrences of polar cosets using the factor graph representation of a polar coset. This approach enabled the selective consideration of relevant polar cosets involved in generating codewords with minimum weight, thereby determining the overall minimum distance and number of occurrences of the entire code. Furthermore, our contributions were extended to compute partial weight spectra of both pure and pre-transformed polar codes.

A notable advantage of this method is its complete independence from the specific code construction and pre-transformation employed. Experimental results underscored

the significantly lower computational complexity of this method compared to previous approaches in the literature.

The third chapter extends the findings of the second chapter to take into account the effects of puncturing and shortening on polar codes distance properties. Introducing the concept of rate-compatible polar cosets, the minimum distance and its associated number of occurrences are computed while considering the impact of puncturing and shortening. By accounting for these transformations, the proposed approach facilitates the computation of the minimum distance and associated occurrences of punctured and shortened (pre-transformed) polar codes, or more generally, their partial weight spectrum. Notably, our approach is flexible and can be adapted to various puncturing or shortening strategies. Once again, numerical results underscore the significantly reduced computational complexity of the proposed method compared to previous techniques in the literature.

In the fourth chapter, the focus shifts towards the requirements regarding list size of the decoding process that is necessary to achieve Maximum Likelihood performance. Initially, it is demonstrated that from a specific decoding index, the Successive Cancellation List decoder can be substituted with a simple Successive Cancellation decoder without incurring any performance loss. Subsequently, constructions are studied in this chapter, revealing that, for short codes, constructions emphasizing the enhancement of polar code distance properties can achieve ML performance under moderate list sizes. However, this observation does not hold for larger block lengths, prompting an exploration of the parameters that impact list size requirements for getting close to optimal code performance.

A key contribution of this chapter in the proposal of an upper bound on the average list size required to achieve ML performance. It is shown that this bound depends on the introduced difference to true path metric of relevant polar cosets at each decoding stage. Numerical results demonstrate that the estimated bound closely aligns with Monte Carlo simulations. Similar to previously proposed methods, this approach can be adapted to different pre-transformations and/or puncturing or shortening patterns. One can note its versatility and applicability across various scenarios.

Perspectives

The various contributions outlined in this manuscript opens up numerous avenues for further research and exploration.

Expanding upon the insights provided in Chapter 2, the proposed method for computing the distance properties of polar cosets is not restricted to polar codes built solely using Arikan's kernel. Rather, it can be applied to a broader class of polar codes that are based on different kernels known as Multikernel polar codes [91]. Multikernel polar codes have been proven to yield the same polarization properties as Arikan's polar codes in [92] when verifying sufficient conditions. The key requirement for applying this method is that the factor graphs that enable the determination of distance properties must have a tree construction, i.e., a cycle-free structure. This condition holds true not only for Arikan's kernel but also for Multikernel constructions where the decoding graph factors exhibit a tree-like structure. Generalizing this approach to Multikernel polar codes offers several benefits. Firstly, it facilitates the estimation of their performance under Maximum Likelihood decoding. Thus, it enables to provide to the code designer valuable insights into code capabilities. Furthermore, it can be leveraged for code construction purposes, enabling the design of Multikernel polar codes optimized for specific performance criteria or application requirements.

In Chapter 4, the methods detailed in Chapters 2 and 3 can be leveraged to optimize code constructions in terms of distance properties. These distance properties are not only crucial for decoding performance but also play a major role in optimizing various parameters associated with polar codes. For instance, the results can be used to optimize parameters such as the CRC generator polynomial in the case of polar codes concatenated with a CRC, or the generator polynomial for Polarization Adjusted Convolutional (PAC) codes. Additionally, the distance properties can inform the optimization of puncturing and shortening patterns and their associated rate-profiles. The low-complexity nature of the proposed algorithm for computing distance properties enhances its utility for optimization tasks. Thus, designers can explore a wide range of parameter configurations.

Actually, the integration of the results from Chapters 2, 3, and 4 provides a complete framework for analyzing and optimizing polar code constructions. By combining these insights, it becomes possible to identify the distance properties of a specific code construction and at the same time establish an upper bound on the required list size for achieving Maximum Likelihood performance. This approach enables the design of polar codes that offer an efficient tradeoff between performance and decoding computational complexity. Consequently, the requirements of different use cases can be achieved. Understanding the underlying factors that influence the required list size for optimal performance and distance properties allows for the construction of codes optimized dedicated to communication scenarios.

Finally, the metric computation method introduced in Chapter 4 in order to compute

the average list size for ML decoding can also be effectively applied for path pruning purposes. In fact, by setting a predefined threshold for the metric, knowing that we are able to have insights on the metric, paths that exceed this threshold can be identified and pruned from consideration during the decoding process. This pruning mechanism helps to reduce the computational complexity of decoding algorithms by eliminating paths that are less likely to correspond to the correct codeword.

Appendices

A Demonstration of equations 2.24 and 2.26

Let $N = 2^n$. $\forall \mathbf{u} \in \{0,1\}^{2N}$, authors of [50] prove that:

$$C_{2N}(\mathbf{u}_{2i}, u_{2i+1}) = \{(\mathbf{c}_1, \mathbf{c}_2) | \mathbf{c}_1 \in C_N(\mathbf{u}_{2i,even} \oplus (\mathbf{u}_{2i,odd}, u_{2i+1})), \mathbf{c}_2 \in C_N(\mathbf{u}_{2i,odd}, u_{2i+1})\} \tag{20}$$

This means that $\forall \mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2) \in C_{2N}(\mathbf{u}_{2i}, u_{2i+1}) | w(\mathbf{c}) = w^*(C_{2N}(\mathbf{u}_{2i}, u_{2i+1}))$ then:

$$\begin{cases} w(\mathbf{c}_1) = w^*(C_N(\mathbf{u}_{2i,even} \oplus (\mathbf{u}_{2i,odd}, u_{2i+1}))) \\ w(\mathbf{c}_2) = w^*(C_N(\mathbf{u}_{2i,odd}, u_{2i+1})) \end{cases} \tag{21}$$

This means that $\forall \mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2) \in C_{2N}(\mathbf{u}_{2i}, u_{2i+1})$ of minimum weight,

$$w(\mathbf{c}) = w^*(C_N(\mathbf{u}_{2i,even} \oplus (\mathbf{u}_{2i,odd}, u_{2i+1})) + w^*(C_N(\mathbf{u}_{2i,odd}, u_{2i+1}))$$

.

In order to build a codeword $\mathbf{c} = (c_1, c_2) \in C_{2N}(\mathbf{u}_{2i}, u_{2i+1})$ with the least weight, we must first select c_1 from $C_N(\mathbf{u}_{2i,even} \oplus (\mathbf{u}_{2i,odd}, u_{2i+1}))$ of minimum weight and c_2 from $C_N(\mathbf{u}_{2i,odd}, u_{2i+1})$ also of minimum weight. if $C_N(\mathbf{u}_{2i,even} \oplus (\mathbf{u}_{2i,odd}, u_{2i+1}))$ contains A_1 codewords with minimum weight and $C_N(\mathbf{u}_{2i,odd}, u_{2i+1})$ contains A_2 codewords with minimum weight then there exists $A_1 A_2$ ways to build $\mathbf{c}$. In other words, the number of codewords of minimum weight contained in $C_{2N}(\mathbf{u}_{2i}, u_{2i+1})$ is $A_1 A_2$. Combining both results amounts exactly to the multiplication of the MWEF, i.e.:

$$A^*_{2N}(\mathbf{u}_{2i}, u_{2i+1}) = A^*_N(\mathbf{u}_{2i,even} \oplus (\mathbf{u}_{2i,odd}, u_{2i+1})) A^*_N((\mathbf{u}_{2i,odd}, u_{2i+1})) \tag{22}$$

$C_{2N}(\mathbf{u}_{2i}, u_{2i+1})$ can be written as:

$$C_{2N}(\mathbf{u}_{2i}, u_{2i+1}) = C_{2N}(\mathbf{u}_{2i}, 0) \bigcup C_{2N}(\mathbf{u}_{2i}, 1) \tag{23}$$

Three different cases may occur:

- The first case is if $w^*(C_{2N}(\mathbf{u}_{2i}, 0)) > w^*(C_{2N}(\mathbf{u}_{2i}, 1))$. In that case:

$$A^*_{2N}(\mathbf{u}_{2i}, u_{2i+1}) = A^*_N(\mathbf{u}_{2i}, 0) = A^*_N(\mathbf{u}_{2i,even} \oplus \mathbf{u}_{2i,odd}) A^*_N(\mathbf{u}_{2i,odd}, 0)$$

  because all the codewords in $\bigcup C_{2N}(\mathbf{u}_{2i}, 1)$ are of strictly higher weight and do not contribute in the minimum weight computation of the polar coset.

- The second case if $w^*(C_{2N}(\mathbf{u}_{2i}, 0)) < w^*(C_{2N}(\mathbf{u}_{2i}, 1))$. Similarly to the first case,

$$A^*_{2N}(\mathbf{u}_{2i}, u_{2i+1}) = A^*_N(\mathbf{u}_{2i}, 1) = A^*_N(\mathbf{u}_{2i,even} \oplus (\mathbf{u}_{2i,odd}, 1)) A^*_N(\mathbf{u}_{2i,odd}, 1)$$

  .

- The third case is if $w^*(C_{2N}(\mathbf{u}_{2i}, 0)) = w^*(C_{2N}(\mathbf{u}_{2i}, 1))$. In this case,

$$A^*_{2N}(\mathbf{u}_{2i}, u_{2i+1}) = A^*_N(\mathbf{u}_{2i}, 0) + A^*_N(\mathbf{u}_{2i}, 1)$$

  . This is explained by the fact that as both cosets contain codewords with minimum weight, summing the monomials $A^*_N(\mathbf{u}_{2i}, 0)$ and $A^*_N(\mathbf{u}_{2i}, 1)$ will take into account the overall number of occurrences of codewords with minimum weight.

If we consider the operator $LP(.)$ that, given a polynomial, only keeps the monomial of least power, the three aforementioned cases can be summarised with:

$$A^*_{2N}(\mathbf{u}_{2i})(X) = LP\Big(\sum_{u_{2i+1} \in \{0,1\}} A^*_N(\mathbf{u}_{2i,\text{even}} \oplus (\mathbf{u}_{2i,\text{odd}}, u_{2i+1}))(X) A^*_N(\mathbf{u}_{2i,\text{odd}}, u_{2i+1})(X) \Big) \tag{24}$$

□

Equation (2.24) can also be deduced when expressing $C_{2N(\mathbf{u}_{2i+1})}$ as [50]:

$$C_{2N}(\mathbf{u}_{2i+1}) = \{(\mathbf{c}_1, \mathbf{c}_2) | \mathbf{c}_1 \in C_N(\mathbf{u}_{2i+1,even} \oplus \mathbf{u}_{2i+1,odd}), \mathbf{c}_2 \in C_N(\mathbf{u}_{2i+1,odd})\} \tag{25}$$

B Proof for equation (2.31) and (2.33)

As for the computation of MWEF, starting from:

$$C_{2N}(\mathbf{u}_{2i}, u_{2i+1}) = \{(\mathbf{c}_1, \mathbf{c}_2) | \mathbf{c}_1 \in C_N(\mathbf{u}_{2i,even} \oplus (\mathbf{u}_{2i,odd}, u_{2i+1})), \mathbf{c}_2 \in C_N(\mathbf{u}_{2i,odd}, u_{2i+1})\}$$

(26)

This means that $\forall \mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2) \in C_{2N}(\mathbf{u}_{2i}, u_{2i+1}) | w(\mathbf{c}) \leq w_{end}$ then:

$$\begin{cases} w(\mathbf{c}_1) \leq w_{end} \\ w(\mathbf{c}_2) \leq w_{end} \\ w(\mathbf{c}_1) + w(\mathbf{c}_1) \leq w_{end} \end{cases} \tag{27}$$

In order to build a codeword $\mathbf{c} = (c_1, c_2) \in C_{2N}(\mathbf{u}_{2i}, u_{2i+1})$ of weight less or equal to w_{end}, we must first select c_1 from $C_N(\mathbf{u}_{2i,even} \oplus (\mathbf{u}_{2i,odd}, u_{2i+1}))$ of weight less or equal to w_{end} and c_2 from $C_N(\mathbf{u}_{2i,odd}, u_{2i+1})$ also of weight less or equal to w_{end} and make sure that $w(\mathbf{c}_1) + w(\mathbf{c}_1) \leq w_{end}$. The number of codewords of weight $w_i \leq w_{end}$ contained in $C_{2N}(\mathbf{u}_{2i}, u_{2i+1})$ is obtained by counting all the possible configurations of c_1 from $C_N(\mathbf{u}_{2i,even} \oplus (\mathbf{u}_{2i,odd}, u_{2i+1}))$ and c_2 from $C_N(\mathbf{u}_{2i,odd}, u_{2i+1})$ such that $w(\mathbf{c}_1) + w(\mathbf{c}_2) = w_i$ Combining both results amounts exactly to the multiplication of the RWEF, i.e.:

$$A_{2N}^{w_{end}}(\mathbf{u}_{2i}, u_{2i+1}) = A_N^{w_{end}}(\mathbf{u}_{2i,even} \oplus (\mathbf{u}_{2i,odd}, u_{2i+1})) A_N^{w_{end}}(\mathbf{u}_{2i,odd}, u_{2i+1}) \tag{28}$$

As $C_{2N}(\mathbf{u}_{2i}, u_{2i+1}) = C_{2N}(\mathbf{u}_{2i}, 0) \bigcup C_{2N}(\mathbf{u}_{2i}, 1)$ then in order to compute the RWEF of $C_{2N}(\mathbf{u}_{2i}, u_{2i+1})$, we have to sum the RWEFs of $C_{2N}(\mathbf{u}_{2i}, 0)$ and $C_{2N}(\mathbf{u}_{2i}, 1)$ and remove the monomials with degrees greater to w_{end}. This leads to:

$$A_{2N}^*(\mathbf{u}_{2i})(X) = LT_{w_{end}}\Big(\sum_{u_{2i+1} \in \{0,1\}} A_N^*(\mathbf{u}_{2i,\text{even}} \oplus (\mathbf{u}_{2i,\text{odd}}, u_{2i+1}))(X) A_N^*(\mathbf{u}_{2i,\text{odd}}, u_{2i+1})(X) \Big) \tag{29}$$

□

Equation (2.24) can also be deduced when using Equation (25)

C Demonstration of equation 4.2

The term $\mathbb{P}\left(\mathbf{U}_0^i = \mathbf{u}_0^i[l] | \mathbf{Y} = \mathbf{y}\right)$ in Equation (4.1) can be formulated as:

$$\mathbb{P}\left(\mathbf{U}_0^i = \mathbf{u}_0^i[l] | \mathbf{Y} = \mathbf{y}\right) = \sum_{\mathbf{u}_{i+1}^{N-1} \in \mathbb{F}_2^{N-i-1}} \mathbb{P}\left(\mathbf{U} = [\mathbf{u}_0^i[l], \mathbf{u}_{i+1}^{N-1}] \Big| \mathbf{y}\right) \tag{30}$$

$$= \frac{\mathbb{P}(\mathbf{U} = \mathbf{u})}{p(\mathbf{y})} \sum_{\mathbf{u}_{i+1}^{N-1} \in \mathbb{F}_2^{N-i-1}} \left(\mathbf{y} | \, \mathbf{U} = [\mathbf{u}_0^i[l], \mathbf{u}_{i+1}^{N-1}]\right) \tag{31}$$

In the case of an AWGN channel, the probability of transition is determined by:

$$p\left(\mathbf{y}|\,\mathbf{u}\right) = \frac{1}{(2\pi)^{n/2}\sigma^n} e^{-\frac{1}{2\sigma^2}||\mathbf{y}-\mathbf{x}(\mathbf{u})||_2^2}$$

where $\mathbf{x}(\mathbf{u}) = 1 - 2\mathbf{c}$ and $\mathbf{c} = \mathbf{u}\mathbf{G}$. Besides, since $\mathbb{P}(\mathbf{U} = \mathbf{u}) = 2^{-N}$ and $p(\mathbf{y})$ is expressed as

$$p(\mathbf{y}) = \sum_{\mathbf{u}} p(\mathbf{y}|\mathbf{u})\mathbb{P}(\mathbf{U} = \mathbf{u})$$

, $\mathbb{P}\left(\mathbf{U}_0^i = \mathbf{u}_0^i|\mathbf{Y} = \mathbf{y}\right)$ is finally expressed as:

$$\mathbb{P}\left(\mathbf{U}_0^i = \mathbf{u}_0^i|\mathbf{Y} = \mathbf{y}\right) = \frac{\sum\limits_{\mathbf{u}_{i+1}^{N-1} \in \mathbb{F}_2^{N-(i+1)}} e^{-\frac{1}{2\sigma^2}||\mathbf{y}-\mathbf{x}(\mathbf{u})||_2^2}}{\sum\limits_{\mathbf{v}\in\{-1,1\}^n} e^{-\frac{1}{2\sigma^2}||\mathbf{y}-\mathbf{v}||_2^2}} \tag{32}$$

Therefore, the path metric $m_i(\mathbf{u}_0^i, \mathbf{y})$ is expressed as:

$$\begin{aligned} m_i(\mathbf{u}_0^i[l], \mathbf{y}) &= \log\left(\sum_{\mathbf{v}\in\{-1,1\}^n} e^{-\frac{1}{2\sigma^2}||\mathbf{y}-\mathbf{v}||_2^2}\right) - \log\left(\sum_{\mathbf{u}_{i+1}^{N-1} \in \mathbb{F}_2^{N-(i+1)}} e^{-\frac{1}{2\sigma^2}||\mathbf{y}-\mathbf{x}(\mathbf{u})||_2^2}\right) \\ &= \max_{\mathbf{v}\in\{-1,1\}^n}{}^{\star}\left(-\frac{1}{2\sigma^2}||\mathbf{y}-\mathbf{v}||_2^2\right) - \max_{\mathbf{u}_{i+1}^{N-1} \in \mathbb{F}_2^{n-(i+1)}}{}^{\star}\left(-\frac{1}{2\sigma^2}||\mathbf{y}-\mathbf{x}(\mathbf{u})||_2^2\right) \end{aligned} \tag{33}$$

Under min-sum approximation of the decoding process, (33) can be simplified by replacing the operator $\max^\star$ by max:

$$\begin{aligned} m_i(\mathbf{u}_0^i[l], \mathbf{y}) &\simeq \max_{\mathbf{v}\in\{-1,1\}^n}\left(-\frac{1}{2\sigma^2}||\mathbf{y}-\mathbf{v}||_2^2\right) - \max_{\mathbf{u}_{i+1}^{N-1} \in \mathbb{F}_2^{n-(i+1)}}\left(-\frac{1}{2\sigma^2}||\mathbf{y}-\mathbf{x}(\mathbf{u})||_2^2\right) \\ &= \frac{1}{2\sigma^2}\min_{\mathbf{u}_{i+1}^{N-1} \in \mathbb{F}_2^{n-(i+1)}}\left(||\mathbf{y}-\mathbf{x}(\mathbf{u})||_2^2\right) - \frac{1}{2\sigma^2}\min_{\mathbf{v}\in\{-1,1\}^n}\left(||\mathbf{y}-\mathbf{v}||_2^2\right) \end{aligned} \tag{34}$$

Finally, $m_i(\mathbf{u}_0^i[l], \mathbf{y})$ can be expressed as :

$$m_i(\mathbf{u}_0^i[l], \mathbf{y}) = \frac{1}{2\sigma^2}\min_{\mathbf{x}\in\mathcal{C}_N(\mathbf{u}_0^i[l])}\left(||\mathbf{y}-\mathbf{x}||_2^2\right) - \frac{1}{2\sigma^2}\min_{\mathbf{v}\in\{-1,1\}^n}\left(||\mathbf{y}-\mathbf{v}||_2^2\right) \tag{35}$$

□

Bibliography

[1] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. on Inf. Theory*, 2009.

[2] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423, 1948. [Online]. Available: http://plan9.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf

[3] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Information and control*, vol. 3, no. 1, pp. 68–79, 1960.

[4] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the society for industrial and applied mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[5] A. Viterbi, "Convolutional codes and their performance in communication systems," *IEEE Transactions on Communication Technology*, vol. 19, no. 5, pp. 751–772, 1971.

[6] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *Proceedings of ICC'93-IEEE International Conference on Communications*, vol. 2. IEEE, 1993, pp. 1064–1070.

[7] R. Gallager, "Low-density parity-check codes," *IRE Transactions on information theory*, vol. 8, no. 1, pp. 21–28, 1962.

[8] I. Tal and A. Vardy, "List decoding of polar codes," in *2021 IEEE International Symposium on Information Theory (ISIT)*, 2011.

[9] 3GPP TS 38.212 V17.4.0, "5G; NR; multiplexing and channel coding," 2023.

[10] E. Arıkan, "From sequential decoding to channel polarization and back again," *arXiv preprint arXiv:1908.09594*, 2019.

[11] P. Trifonov and V. Miloslavskaya, "Polar codes with dynamic frozen symbols and their decoding by directed search," in *2013 IEEE Information Theory Workshop (ITW)*, 2013, pp. 1–5.

[12] S. Gelincik, P. Mary, A. Savard, and J.-Y. Baudais, "Preserving the minimum distance of polar-like codes while increasing the information length," in *2022 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2022, pp. 2130–2135.

[13] B. Shuval and I. Tal, "Strong polarization for shortened and punctured polar codes," *arXiv preprint arXiv:2401.16833*, 2024.

[14] V. Miloslavskaya, B. Vucetic, and Y. Li, "Computing the partial weight distribution of punctured, shortened, precoded polar codes," *IEEE Transactions on Communications*, 2022.

[15] C. Leroux, A. J. Raymond, G. Sarkis, and W. J. Gross, "A semi-parallel successive-cancellation decoder for polar codes," *IEEE Transactions on Signal Processing*, vol. 61, no. 2, pp. 289–299, 2012.

[16] M. Bardet, V. Dragoi, A. Otmani, and J.-P. Tillich, "Algebraic properties of polar codes from a new polynomial formalism," in *IEEE ISIT*, 2016.

[17] C. Schürch, "A partial order for the synthesized channels of a polar code," in *2016 IEEE International Symposium on Information Theory (ISIT)*, 2016, pp. 220–224.

[18] R. Mori and T. Tanaka, "Performance of polar codes with the construction using density evolution," *IEEE Communications Letters*, vol. 13, no. 7, pp. 519–521, 2009.

[19] P. Trifonov, "Efficient design and decoding of polar codes," *IEEE transactions on communications*, vol. 60, no. 11, pp. 3221–3227, 2012.

[20] D. E. Muller, "Application of boolean algebra to switching circuit design and to error detection," *Transactions of the I.R.E. Professional Group on Electronic Computers*, vol. EC-3, no. 3, pp. 6–12, 1954.

[21] J. Dai, K. Niu, Z. Si, C. Dong, and J. Lin, "Does gaussian approximation work well for the long-length polar code construction?" *IEEE Access*, vol. 5, pp. 7950–7963, 2017.

[22] B. Li, H. Shen, and D. Tse, "A rm-polar codes," *arXiv preprint arXiv:1407.5483*, 2014.

[23] N. Hussami, S. B. Korada, and R. Urbanke, "Performance of polar codes for channel and source coding," in *IEEE ISIT*, 2009.

[24] B. Li, H. Shen, and D. Tse, "An adaptive successive cancellation list decoder for polar codes with cyclic redundancy check," *IEEE Communications Letters*, 2012.

[25] B. Li, H. Zhang, and J. Gu, "On pre-transformed polar codes," 2019. [Online]. Available: arXiv2019,arXiv:1912.06359

[26] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg, "Llr-based successive cancellation list decoding of polar codes," *IEEE Transactions on Signal Processing*, vol. 63, no. 19, pp. 5165–5179, 2015.

[27] H. Yao, A. Fazeli, and A. Vardy, "List decoding of arkans pac codes," *Entropy*, vol. 23, no. 7, 2021. [Online]. Available: https://www.mdpi.com/1099-4300/23/7/841

[28] J. M. Wozencraft, "Sequential decoding for reliable communication," 1957. [Online]. Available: https://api.semanticscholar.org/CorpusID:14879562

[29] F. Jelinek, "Fast sequential decoding algorithm using a stack," *IBM Journal of Research and Development*, vol. 13, no. 6, pp. 675–685, 1969.

[30] R. Fano, "A heuristic discussion of probabilistic decoding," *IEEE Transactions on Information Theory*, vol. 9, no. 2, pp. 64–74, 1963.

[31] K. Niu and K. Chen, "Stack decoding of polar codes," *Electronics Letters*, vol. 48, pp. 695–697, 2012. [Online]. Available: https://api.semanticscholar.org/CorpusID:117052030

[32] K. Chen, K. Niu, and J. Lin, "Improved successive cancellation decoding of polar codes," *IEEE Transactions on Communications*, vol. 61, no. 8, pp. 3100–3107, 2013.

[33] V. Miloslavskaya and P. Trifonov, "Sequential decoding of polar codes," *IEEE Communications Letters*, vol. 18, no. 7, pp. 1127–1130, 2014.

[34] G. Trofimiuk, N. Iakuba, S. Rets, K. Ivanov, and P. Trifonov, "Fast block sequential decoding of polar codes," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 10, pp. 10 988–10 999, 2020.

[35] Y. Polyanskiy, H. V. Poor, and S. Verdu, "Channel coding rate in the finite blocklength regime," *IEEE Transactions on Information Theory*, vol. 56, no. 5, pp. 2307–2359, 2010.

[36] M. Rowshan, A. Burg, and E. Viterbo, "Polarization-adjusted convolutional (pac) codes: Sequential decoding vs list decoding," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 2, pp. 1434–1447, 2021.

[37] M. Moradi, "On sequential decoding metric function of polarization-adjusted convolutional (pac) codes," *IEEE Transactions on Communications*, vol. 69, no. 12, pp. 7913–7922, 2021.

[38] G. Vazquez-Vilar, A. G. i Fabregas, T. Koch, and A. Lancho, "Saddlepoint approximation of the error probability of binary hypothesis testing," in *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2018, pp. 2306–2310.

[39] J. Font-Segura, G. Vazquez-Vilar, A. Martinez, A. G. i Fàbregas, and A. Lancho, "Saddlepoint approximations of lower and upper bounds to the error probability in channel coding," in *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 2018, pp. 1–6.

[40] G. Durisi and A. Lancho, "Transmitting short packets over wireless channelsan information-theoretic perspective," 2020, https://gdurisi.github.io/fbl-notes/.

[41] M. C. Cokun, G. Durisi, T. Jerkovits, G. Liva, W. E. Ryan, B. T. Stein, and F. Steiner, "Efficient error-correcting codes in the short blocklength regime," *Phys. Commun.*, vol. 34, pp. 66–79, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:56517208

[42] T. Tonnellier and W. J. Gross, "On systematic polarization-adjusted convolutional (pac) codes," *IEEE Communications Letters*, vol. 25, no. 7, pp. 2128–2132, 2021.

[43] M.-C. Chiu and Y.-S. Su, "Design of polar codes and pac codes for scl decoding," *IEEE Transactions on Communications*, vol. 71, no. 5, pp. 2587–2601, 2023.

[44] S. K. Mishra, D. Katyal, and S. A. Ganapathi, "A heuristic algorithm for rate-profiling of polarization adjusted convolutional (pac) codes," Oct. 2021. [Online]. Available: http://dx.doi.org/10.36227/techrxiv.16735351.v1

[45] M. C. Cokun and H. D. Pfster, "An information-theoretic perspective on successive cancellation list decoding and polar code design," *IEEE Transactions on Information Theory*, vol. 68, no. 9, pp. 5779–5791, 2022.

[46] V. Miloslavskaya, L. Yonghui, and B. Vucetic, "Frozen set design for precoded polar codes," 2023. [Online]. Available: arXiv:2311.10047

[47] M. Moradi, A. Mozammel, K. Qin, and E. Arikan, "Performance and complexity of sequential decoding of PAC codes," *CoRR*, 2020.

[48] T. Tonnellier and W. J. Gross, "On systematic polarization-adjusted convolutional (pac) codes," *IEEE Communications Letters*, 2021.

[49] R. Polyanskaya, M. Davletshin, and N. Polyanskii, "Weight distributions for successive cancellation decoding of polar codes," *IEEE Transactions on Communications*, vol. 68, no. 12, pp. 7328–7336, 2020.

[50] H. Yao, A. Fazeli, and A. Vardy, "A deterministic algorithm for computing the weight distribution of polar code," *IEEE Transactions on Information Theory*, 2023.

[51] J. Piao, K. Niu, J. Dai, and C. Dong, "Sphere constraint based enumeration methods to analyze the minimum weight distribution of polar codes," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 10, pp. 11 557–11 569, 2020.

[52] M. Valipour and S. Yousefi, "On probabilistic weight distribution of polar codes," *IEEE communications letters*, vol. 17, no. 11, pp. 2120–2123, 2013.

[53] Q. Zhang, A. Liu, and X. Pan, "An enhanced probabilistic computation method for the weight distribution of polar codes," *IEEE Communications Letters*, vol. 21, no. 12, pp. 2562–2565, 2017.

[54] Y. Li, H. Zhang, R. Li, J. Wang, G. Yan, and Z. Ma, "On the weight spectrum of pre-transformed polar codes," in *2021 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2021, pp. 1224–1229.

[55] M. Rowshan, V.-F. Drăgoi, and J. Yuan, "On the closed-form weight enumeration of polar codes: 1.5d-weight codewords," *arXiv preprint arXiv:2305.02921*, 2023.

[56] Z. Ye, Y. Li, H. Zhang, J. Wang, G. Yan, and Z. Ma, "On the distribution of weights less than 2w min in polar codes," *IEEE Transactions on Communications*, 2024.

[57] M. Rowshan, V.-F. Drăgoi, and J. Yuan, "Weight structure of low/high-rate polar codes and its applications," *arXiv preprint arXiv:2402.12707*, 2024.

[58] M. Rowshan and J. Yuan, "Fast enumeration of minimum weight codewords of PAC codes," in *IEEE ITW*, 2022.

[59] A. Zunker, M. Geiselhart, and S. Ten Brink, "Enumeration of minimum weight codewords of pre-transformed polar codes by tree intersection," in *2024 58th Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 2024, pp. 1–6.

[60] T. Richardson and R. Urbanke, *Modern Coding Theory*, Chapter 2, Cambridge University Press, 2008.

[61] R. Mori and T. Tanaka, "Performance and construction of polar codes on symmetric binary-input memoryless channels," in *IEEE ISIT*, 2009.

[62] I. Tal and A. Vardy, "How to construct polar codes," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6562–6582, 2013.

[63] P. Trifonov, "Efficient design and decoding of polar codes," *IEEE Transactions on Communications*, 2012.

[64] https://github.com/mohammad-rowshan/Fast-Enumeration-of- Minimum-Weight-Codewords-of-PAC-Codes.

[65] M. Rowshan and J. Yuan, "On the minimum weight codewords of pac codes: The impact of pre-transformation," *IEEE Journal on Selected Areas in Information Theory*, 2023.

[66] L. Zhang, Z. Zhang, X. Wang, Q. Yu, and Y. Chen, "On the puncturing patterns for punctured polar codes," in *2014 IEEE International Symposium on Information Theory*, 2014, pp. 121–125.

[67] J. Kim, J.-H. Kim, and S.-H. Kim, "An efficient search on puncturing patterns for short polar codes," in *2015 International Conference on Information and Communication Technology Convergence (ICTC)*, 2015, pp. 182–184.

[68] K. Niu, K. Chen, and J.-R. Lin, "Beyond turbo codes: Rate-compatible punctured polar codes," in *2013 IEEE International Conference on Communications (ICC)*, 2013, pp. 3423–3427.

[69] V. Bioglio, F. Gabry, and I. Land, "Low-complexity puncturing and shortening of polar codes," in *2017 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, 2017, pp. 1–6.

[70] K. Niu, J. Dai, K. Chen, J. Lin, Q. Zhang, and A. V. Vasilakos, "Rate-compatible punctured polar codes: Optimal construction based on polar spectra," *arXiv preprint arXiv:1612.01352*, 2016.

[71] L. Chandesris, V. Savin, and D. Declercq, "On puncturing strategies for polar codes," in *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2017, pp. 766–771.

[72] D.-M. Shin, S.-C. Lim, and K. Yang, "Design of length-compatible polar codes based on the reduction of polarizing matrices," *IEEE Transactions on Communications*, vol. 61, no. 7, pp. 2593–2599, 2013.

[73] G. Sarkis, I. Tal, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, "Flexible and low-complexity encoding and decoding of systematic polar codes," *IEEE Transactions on Communications*, vol. 64, no. 7, pp. 2732–2745, 2016.

[74] R. Wang and R. Liu, "A novel puncturing scheme for polar codes," *IEEE Communications Letters*, vol. 18, no. 12, pp. 2081–2084, 2014.

[75] V. Miloslavskaya, "Shortened polar codes," *IEEE Transactions on Information Theory*, vol. 61, no. 9, pp. 4852–4865, 2015.

[76] X. Gu, M. Rowshan, and J. Yuan, "Rate-compatible shortened pac codes," in *2023 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*, 2023, pp. 1–6.

[77] M. Ellouze, R. Tajan, C. Leroux, C. Jégo, and C. Poulliat, "Low-complexity algorithm for the minimum weight distribution of polar codes." *IEEE Communications Letters (submitted)*. [Online]. Available: https://filesender.renater.fr/?s=download&token=de39230e-6748-4e5f-b1e8-e7669a888eb1

[78] M. Moradi and A. Mozammel, "A monte-carlo based construction of polarization-adjusted convolutional (pac) codes," *arXiv preprint arXiv:2106.08118*, 2021.

[79] A. Elkelesh, M. Ebada, S. Cammerer, and S. Ten Brink, "Decoder-tailored polar code design using the genetic algorithm," *IEEE Transactions on Communications*, vol. 67, no. 7, pp. 4521–4534, 2019.

[80] V. Miloslavskaya, B. Vucetic, Y. Li, G. Park, and O.-S. Park, "Recursive design of precoded polar codes for scl decoding," *IEEE transactions on communications*, vol. 69, no. 12, pp. 7945–7959, 2021.

[81] M. Moradi and D. G. Mitchell, "Pac code rate-profile design using search-constrained optimization algorithms," *arXiv preprint arXiv:2401.10376*, 2024.

[82] S. Gelincik, P. Mary, J.-Y. Baudais, and A. Savard, "Achieving pac code performance with scl decoding without extra computational complexity," in *ICC 2022-IEEE International Conference on Communications*. IEEE, 2022, pp. 104–109.

[83] A. Zunker, M. Geiselhart, L. Johannsen, C. Kestel, S. t. Brink, T. Vogt, and N. Wehn, "Row-merged polar codes: Analysis, design and decoder implementation," *arXiv preprint arXiv:2312.14749*, 2023.

[84] Z. Zhang, L. Zhang, X. Wang, C. Zhong, and H. V. Poor, "A split-reduced successive cancellation list decoder for polar codes," *IEEE J. Sel. Areas in Commun.*, vol. 34, no. 2, pp. 292–302, 2016.

[85] L. Zhang, A. Cao, J. Qiao, and Y. He, "A crc-aided sr-scl decoding algorithm for polar codes," in *2019 IEEE MTT-S International Wireless Symposium (IWS)*, 2019, pp. 1–3.

[86] M. Moradi and A. Mozammel, "A tree pruning technique for decoding complexity reduction of polar codes and pac codes," *IEEE Transactions on Communications*, 2023.

[87] M. Rowshan and E. Viterbo, "Stepped list decoding for polar codes," in *2018 IEEE 10th International Symposium on Turbo Codes & Iterative Information Processing (ISTC)*. IEEE, 2018, pp. 1–5.

[88] A. Fazeli, A. Vardy, and H. Yao, "List decoding of polar codes: How large should the list be to achieve ml decoding?" in *2021 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2021, pp. 1594–1599.

[89] Y. L. Tong, "The multivariate normal distribution," 1989. [Online]. Available: https://api.semanticscholar.org/CorpusID:125040417

[90] M. Rowshan, S. H. Dau, and E. Viterbo, "On the formation of min-weight codewords of polar/pac codes and its applications," *IEEE Transactions on Information Theory*, 2023.

[91] F. Gabry, V. Bioglio, I. Land, and J.-C. Belfiore, "Multi-kernel construction of polar codes," in *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2017, pp. 761–765.

[92] M. Benammar, V. Bioglio, F. Gabry, and I. Land, "Multi-kernel polar codes: Proof of polarization and error exponents," in *2017 IEEE Information Theory Workshop (ITW)*. IEEE, 2017, pp. 101–105.