Grenoble INP – ENSIMAG

École Nationale Supérieure d'Informatique et de Mathématiques Appliquées

Rapport de Projet de Fin d'Études

Effectué au Loria

Laboratoire lorrain de recherche en informatique et ses applications

Topological Concatenation for 2D Quantum Error Correction

Alexandre Guernut
3ème année – Option MMIS

01 mars 2021 – 27 août 2021

Loria
615 rue du Jardin Botanique
54600 Villers-lès-Nancy
France

Responsable de stage
Christophe Vuillot
Tuteur de l'école
Matthieu Chabanas

Table des matières

Résumé

Les qubits des ordinateurs quantiques sont sujets aux erreurs. Pour améliorer la situation, les qubits sont encodés grâce à des codes quantiques correcteurs d'erreurs, ressemblant beaucoup à leurs homologues classiques. Lors de l'exécution de l'algorithme quantique, on réalise de façon régulière des étapes de correction d'erreurs.

Afin d'augmenter la fiabilité de certains codes quantiques, une technique dite de concaténation permet d'abaisser la probabilité d'erreurs, en encodant récursivement les qubits logiques avec d'autres qubits logiques [1].

D'autres types de codes quantiques correcteurs d'erreurs existent et brillent par leur disposition simple dans l'espace : les codes topologiques. Parmi eux, certains (les codes couleur) sont des candidats naturels sur lesquels appliquer les techniques de concaténation [2].

L'objectif de ce stage est la construction et l'étude d'une procédure de concaténation dans le cas particulier des codes de couleur en deux dimensions.

Mots-clés : *Concatenation of quantum codes*, *Topological codes*, *2D Quantum-error correction*, *Quantum noise*, *Color codes*

Introduction

L'ordinateur et le calcul quantiques offrent des perspectives pour la résolution de problèmes génériques intervenant dans de nombreux domaines industriels (optimisation, chimie, ...) auxquels l'informatique classique ne semble pas être en mesure de répondre de façon efficace. Ces possibilités s'ouvrent au prix d'un accroissement de la complexité de la programmation et de la réalisation de l'architecture exécutant les algorithmes. En particulier, cette dernière est (entre autres problèmes) très sujette aux erreurs et sensible au bruit, qu'il convient de corriger afin d'espérer obtenir une exécution correcte.

Outre la diminution drastique du bruit et des erreurs au sein des ordinateurs quantiques, la possibilité de créer des machines tolérantes à ces erreurs (les détectant et les corrigeant à la volée) est à l'étude. Deux approches se distinguent : une adaptation des codes de théorie de l'information classique – théoriquement plus performants mais dont l'implémentation pratique pose problème – et une approche dite topologique, satisfaisant bien les contraintes physiques et donc plus facilement implémentables.

Il est légitime de se demander si une solution hybride – utilisant des codes topologiques concaténés selon certains motifs fractals – pourrait éventuellement être intéressante et disposer de meilleures propriétés. Au vu des coûts de production de circuits quantiques, la simulation demeure la meilleure façon de quantifier les gains de performances éventuels.

Ce rapport s'articule suivant le plan proposé : après avoir présenté le contexte et la problématique de ce stage, je présenterai les solutions étudiées et pistes envisagées, ainsi que les résultats obtenus. La lecture de la version PDF est conseillée afin de bénéficier des hyperliens permettant de naviguer au sein de celle-ci et d'un meilleur contraste.

1 Présentation de la structure d'accueil

1.1 Présentation du laboratoire

Le Loria (Laboratoire lorrain de recherche en informatique et ses applications) est une Unité mixte de Recherche créée en 1997, confluence des efforts de recherche en informatique du CNRS, de l'Université de Lorraine et de l'Inria en région Lorraine. Sa vocation première est la recherche fondamentale et appliquée en informatique, notamment autour des thématiques de l'image, des réseaux, du traitement numérique du langage, de l'intelligence artificielle et de la robotique. De nombreux logiciels sont nés au Loria, dont la bibliothèque de calcul à virgule flottante GNU MPFR et le langage de programmation par filtrage Tom.

Le laboratoire est situé sur le campus universitaire de Nancy (Fig. 1). La situation sanitaire étant ce qu'elle est, le télé-travail a été systématique durant ce stage. Le logiciel Mattermost, comparable à Slack permet d'échanger régulièrement sur les projets en cours et de façon plus informelle. La plateforme Gitlab permet le partage et la collaboration sur les différents projets de développement. BigBlueButton s'est avéré être un outil utile pour les visio-conférences nécessitant un partage d'écran.

Les différents travaux de recherche sont répartis entre plusieurs départements regroupant chacun plusieurs équipes. Certaines d'entre elles sont des équipes-projet Inria et possèdent une organisation horizontale du point de vue académique, malgré la présence d'un responsable administratif choisi parmi les chercheurs.

Figure 1 – Site du Loria (Vandœuvre-lès-Nancy)

1.2 Présentation de l'équipe MOCQUA

L'équipe du Loria qui m'a accueilli est l'équipe MOCQUA (MOdèles de Calcul classiques et QUAntiques), appartenant au département de Méthodes formelles. Ses principaux axes de

recherche sont la théorie de la complexité et de la calculabilité pour le calcul à l'ordre supérieur, la tolérance aux fautes dans le cadre des systèmes dynamiques et enfin le calcul quantique. L'objectif principal de l'équipe est de répondre aux défis posés par les modèles de calcul nouveaux et futurs. Cette équipe se démarque par son intérêt particulier pour les modèles de calcul ne se réduisant pas à des suites finies de bits.

L'équipe comporte dix chercheurs permanents ainsi que des doctorants et post-doctorants. Christophe Vuillot, mon responsable de stage, l'a rejointe au début de l'année à l'issue de son contrat de post-doctorant à l'Inria de Paris. Emmanuel Jeandel assure le rôle de responsable de l'équipe. Vladimir Zamdzhiev organise des séminaires hebdomadaires sur différents sujets connexes aux centres d'intérêts de l'équipe lors desquels un chercheur vient présenter certains de ses résultats. J'ai moi même présenté mon stage lors d'un de ces séminaires, en prévision de ma présentation lors du *Workshop on Quantum Resource Estimation* organisé au sein de l'*International Symposium on Computer Architecture (ISCA) 2021*.

Bien que comportant quelques chercheurs venus de l'étranger, l'équipe se compose majoritairement de chercheurs issus de grandes écoles françaises (ENS Lyon, Mines de Nancy, Ensimag...).

Le sujet du stage porte sur le calcul quantique tolérant aux fautes, sujet de la thèse de Christophe. En particulier, il vise à étudier les performances d'un code quantique issu d'une déformation d'un code couleur [2].

2 Problématique du stage

Le sujet et la nature du stage ne permettent pas d'allier les critères de concision et d'appréhensibilité par un non-spécialiste. Comme une part non négligeable du stage a consisté en la découverte et la compréhension des notions inhérentes à la problématique, il ne me paraît pas hors de propos de poser certains fondements nécessaires pour décrire avec une précision appréciable la problématique de ce stage. Le cheminement retenu reprend en partie celui que [3] réalise de façon plus exhaustive, complété par certaines notions spécifiques introduites en [1] et [4].

2.1 Cadre et modèle mathématiques

2.1.1 Qubits et opérateurs

Les axiomes de la physique quantique imposent que l'état physique d'un système puisse être décrit par un vecteur (ket) unitaire d'un espace de Hilbert complexe. Dans le cadre restreint de l'informatique quantique, on considère généralement des espaces de Hilbert de dimension finie, et donc disposant d'une base. Afin de manipuler de l'information, par analogie avec l'informatique classique, on utilise des qubits (*quantum bits*). Un qubit est décrit par un espace de Hilbert $\mathcal{H}$ de dimension 2, dont une base orthonormée canonique est :

$$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \qquad |1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$$

L'état d'un qubit est donc décrit par une combinaision linéaire complexe de ces états :

$$|\varphi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \begin{pmatrix}\alpha\\\beta\end{pmatrix} \quad \forall\,\alpha,\beta \in \mathbb{C} \text{ tels que } |\alpha|^2 + |\beta|^2 = 1$$

Cette condition impose bel et bien que le qubit soit de norme unitaire, mais laisse la phase globale indéfinie. On considère que $|\varphi\rangle$ et $e^{i\theta}\,|\varphi\rangle$ représentent le même état, puisque cette phase globale n'est pas physiquement significative. Ainsi, toutes les opérations sur les qubits (et les corrections d'erreurs) pourront être considérées exactes si elles sont justes à une phase globale près.

Un autre axiome de la physique quantique postule que toute évolution temporelle d'un état physique (mesure mise à part) peut être décrite au moyen d'un opérateur unitaire de l'espace de Hilbert associé. Outre la matrice identité, on peut citer parmi les plus importants :

$$X = \begin{pmatrix}0&1\\1&0\end{pmatrix} \quad Y = i\begin{pmatrix}0&-1\\1&0\end{pmatrix} \quad Z = \begin{pmatrix}1&0\\0&-1\end{pmatrix} \quad H = \frac{1}{\sqrt{2}}\begin{pmatrix}1&1\\1&-1\end{pmatrix}$$

L'opérateur X est souvent appelé *bit-flip operator* car il agit comme une porte classique NOT :

$$X\,|\varphi\rangle = \begin{pmatrix}0&1\\1&0\end{pmatrix}\begin{pmatrix}\alpha\\\beta\end{pmatrix} = \begin{pmatrix}\beta\\\alpha\end{pmatrix} \quad \text{En particulier, } X\,|0\rangle = |1\rangle \text{ et } X\,|1\rangle = |0\rangle\,.$$

L'opérateur Z est quant à lui souvent appelé *phase-flip operator* car il inverse la différence de phase dans l'écriture de l'état du qubit :

$$Z\,|\varphi\rangle = \begin{pmatrix}1&0\\0&-1\end{pmatrix}\begin{pmatrix}\alpha\\\beta\end{pmatrix} = \begin{pmatrix}\alpha\\-\beta\end{pmatrix} \quad \text{En particulier, } Z\,|0\rangle = |0\rangle \text{ et } Z\,|1\rangle = -\,|1\rangle\,.$$

Le déroulement d'un calcul quantique sur un qubit fait partie des évolutions temporelles possibles de ce qubit, et donc est nécessairement descriptible par un opérateur unitaire. Bien qu'on pourrait imaginer implémenter n'importe quel opérateur, on préfère se restreindre à un sous-ensemble universel d'opérateurs, dont l'application successive est capable de réaliser tous les opérateurs possibles. Par exemple, on peut remarquer :

$$XZ\left|0\right\rangle = X\left|0\right\rangle = \left|1\right\rangle = -iY\left|0\right\rangle \text{ et } XZ\left|1\right\rangle = -X\left|1\right\rangle = -\left|0\right\rangle = -iY\left|1\right\rangle$$

Ainsi, pour tous les vecteurs de la base, l'application de l'opérateur Z puis de l'opérateur X induit le même effet qu'une application de l'opérateur Y, à une **même** phase près. Comme ce comportement est identique sur toute la base de $\mathcal{H}$, il en est de même par linéarité sur tout état $\left|\varphi\right\rangle$ et donc il est suffisant de savoir appliquer X et Z pour pouvoir appliquer Y.

Il est cependant limitant de considérer des systèmes comportant un seul qubit. En effet, il est nécessaire de pouvoir manipuler plusieurs qubits pour espérer réaliser des choses utiles.

Un système à 2 qubits peut être décrit par un vecteur évoluant dans un espace de dimension $4 = 2^2$, couramment noté $\mathcal{H} \otimes \mathcal{H}$ ou $\mathcal{H}^{\otimes 2}$. Il s'agit du produit tensoriel de $\mathcal{H}$ avec lui même. Une base pour cet espace s'obtient en prenant le produit tensoriel d'une base de $\mathcal{H}$ avec elle-même :

$$\left|00\right\rangle = \left|0\right\rangle \otimes \left|0\right\rangle = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}, \left|01\right\rangle = \left|0\right\rangle \otimes \left|1\right\rangle = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}, \left|10\right\rangle = \left|1\right\rangle \otimes \left|0\right\rangle = \begin{pmatrix}0\\0\\1\\0\end{pmatrix}, \left|11\right\rangle = \left|1\right\rangle \otimes \left|1\right\rangle = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

De même, lorsque l'on considère n qubits, on manipule des vecteurs de $\mathcal{H}^{\otimes n}$. Ces espaces disposent également d'opérateurs, dont certains sont exprimables par des produits tensoriels d'opérateurs sur un qubit. Par exemple :

$$(Z \otimes X)(\alpha\left|\phi\right\rangle \otimes \beta\left|\psi\right\rangle) = \alpha\beta(Z\left|\phi\right\rangle \otimes X\left|\psi\right\rangle)$$

L'opérateur $Z \otimes X$ sur $\mathcal{H}^{\otimes 2}$ applique Z sur le premier qubit et X sur le second. On note généralement ces opérateurs dans $\mathcal{H}^{\otimes n}$ plus simplement $X_I Y_J Z_K$ où I, J, K sont des sous-ensembles disjoints d'indices de $[1, \ldots, n]$ désignant les qubits sur lesquels sont appliqués respectivement les opérateurs X, Y, Z. $Z \otimes X$ est simplement noté $X_2 Z_1$, $Z \otimes Z \otimes I \otimes Z$ simplement $Z_{\{1,2,4\}}$ ou $Z_1 Z_2 Z_4$. Néanmoins, les opérateurs sur plusieurs qubits ne sont pas tous exprimables comme produits tensoriels d'opérateurs sur un qubit. Le plus connu est certainement :

$$CNOT = \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix} \quad \begin{array}{ll} CNOT\left|00\right\rangle = \left|00\right\rangle & CNOT\left|01\right\rangle = \left|01\right\rangle \\ CNOT\left|10\right\rangle = \left|11\right\rangle & CNOT\left|11\right\rangle = \left|10\right\rangle \end{array}$$

Son effet est simplement d'appliquer X sur le second qubit lorsque le premier est à $\left|1\right\rangle$. Il introduit des dépendances entre les qubits – des intrications, nécessaires pour la réalisation d'opérations complexes.

2.1.2 Observables et mesures

Savoir manipuler des qubits au cours d'un calcul ne suffit pas : il faut pouvoir donner du sens au résultat final du calcul. L'idée selon laquelle on accède à 2^n bits d'information en mesurant n qubits est fausse. L'espace $\mathcal{H}^{\otimes n}$ est de dimension 2^n ce qui implique qu'il y a bel et bien un gain en terme de complexité des opérations réalisables avec les qubits. Cependant, il n'est pas physiquement possible d'accéder à toutes ces informations par une unique mesure, celle-ci répondant à certaines spécificités.

Pour un qubit $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$ quelconque, il n'est pas possible d'accéder par une simple mesure aux valeurs α et β. Les axiomes de la physique quantique limitent ce qui est mesurable. Ainsi, il n'est possible de mesurer que les valeurs propres de certains opérateurs – les observables – dont les principaux exemples sont X, Y et Z. Rappelons ici que :

$$Z|0\rangle = |0\rangle \quad \text{Donc } |0\rangle \text{ est un vecteur propre de } Z \text{ associé à la valeur propre 1.}$$

$$Z|1\rangle = -|1\rangle \quad \text{Donc } |1\rangle \text{ est un vecteur propre de } Z \text{ associé à la valeur propre -1.}$$

Pour un état quelconque décomposé dans la base canonique (qui se trouve être la base propre orthonormale associée à Z) $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$, la physique quantique prédit que la mesure de l'observable Z donne la valeur propre 1 (associée à $|0\rangle$) avec une probabilité $|\alpha|^2$ et la valeur propre -1 (associée à $|1\rangle$) avec une probabilité $|\beta|^2$. La condition d'unitarité de l'état du qubit assure la bonne définition de ce principe de mesure. Par ailleurs, une fois la mesure effectuée, l'état quantique du qubit s'effondre (*collapse*) vers l'état propre dont on a mesuré la valeur propre, ce qui impose que toute mesure pratiquée ultérieurement donne le même résultat. Pour en apprendre plus sur les valeurs de α et β, il faut soit répéter la mesure en recalculant et mesurant l'état $|\varphi\rangle$ un grand nombre de fois, soit le simuler classiquement, ce qui devient pratiquement impossible au delà de cinquante qubits.

2.1.3 Modèles d'erreurs et calcul tolérant aux fautes

Les erreurs sont généralement dues à l'interaction spontanée du qubit avec l'extérieur (il n'est pas un système parfaitement isolé) ou à des défauts d'application de portes quantiques. Elles peuvent se modéliser par l'application inopinée d'un opérateur inconnu sur l'état du système. Pour des raisons détaillées dans [3], il est suffisant de savoir corriger l'application inopinée de X ou (inclusif) de Z sur un qubit pour pouvoir être capable de corriger toute la classe d'erreurs possibles qui nous intéresse. De plus, on supposera indépendantes les erreurs survenant sur différents qubits.

Suivant le modèle d'erreurs choisi, les erreurs peuvent apparaître à différents moments lors du calcul. Dans le modèle le plus simple, on considère que la préparation des qubits ainsi que les mesures d'observables sont parfaitement réalisées. Les erreurs n'ont alors lieu que lors de l'application fautive d'une porte, ou lors du stockage bruité d'un qubit dans l'attente d'être manipulé. Pour le modèle le plus élaboré, les mesures et la préparation des états quantiques ne sont plus parfaites, ce qui complexifie la procédure.

Toutes les sources d'erreurs ne sont pas défaillantes avec la même probabilité. Il est assez courant de considérer que c'est le cas, en prenant pour probabilité d'erreur générique p_e celle de la procédure la plus défaillante.

L'objectif du calcul tolérant aux fautes est de trouver une procédure d'encodage et de décodage des qubits, de sorte que l'application régulière d'une étape de correction d'erreurs assure qu'à tout instant, la probabilité que le système comporte un nombre incorrectible d'erreurs soit dominée par p_e^2 (pour les codes corrigeant une erreur) et donc réduite par rapport à p_e.

2.2 Correction d'erreurs

2.2.1 Codes linéaires classiques

La théorie classique de l'information propose des codes permettant un transport fiable de l'information, corrigeant un certain nombre d'erreurs lorsque le canal est bruité. L'idée de code la plus simple est celle du code à trois répétitions, corrigeant une erreur. Au lieu d'envoyer un bit d'information au travers d'un canal soumis aux erreurs, on en transmet trois copies et l'on procède à un vote pour décider de la valeur la plus probable du bit envoyé. Si chaque copie du

bit est inversée avec une probabilité $p_e < \frac{1}{2}$, la valeur la plus présente est la valeur du bit la plus probable.

Bien que séduisante, cette solution est peu efficace et même impossible à adapter en quantique car il n'est pas possible de cloner un qubit. Néanmoins, ce code montre l'idée générale de tout code correcteur d'erreurs : introduire de la redondance d'information pour pouvoir recouvrer l'état initial.

D'autres codes classiques linéaires plus performant existent. Leur but est de coder k bits d'information (soit 2^k messages possibles) en n bits transmis (2^n valeurs possibles dont 2^k mots de code valides). Pour ce faire, on se dote d'une matrice $k \times n$ de rang k, appelée matrice génératrice G. Le mot de code associé au message m est alors $G^T m$. Le code est également associé à une matrice de parité de rang $n-k$, de sorte que $HG^T = 0$. Ainsi, pour chaque message reçu x, la quantité Hx est le vecteur nul si x est un mot du code et un vecteur non nul autrement, indiquant que des bits ont été inversés. Le nombre d'erreurs correctibles ainsi que le processus de décodage dépendent du code utilisé.

Deux propriétés importantes (illustrées par l'exemple du code de Hamming) sont à noter car elles sont exploitées par les codes quantiques de correction d'erreurs. D'une part, la possibilité de détecter les erreurs grâce à une matrice de parité qui donne par la même occasion une méthode pour les corriger. D'autre part, les mots de code sont des combinaisons linéaires des colonnes de G.

2.2.2 Codes stabilisateurs

Pour un code quantique, on veut pouvoir stocker l'état d'un qubit logique (objet mathématique) en encodant ce dernier sur plusieurs qubits physiques (objets réels tels des ions piégés par exemple). Ce sont ces qubits logiques qui seront manipulés lors de l'exécution des algorithmes quantiques. Chaque porte (ou opérateur) logique sera réalisée par l'application de différentes portes sur les qubits physiques, de sorte à obtenir un autre mot du code. Les qubits physiques seront constamment sujets aux erreurs, il faudra donc aussi les corriger au fur et à mesure pour que les erreurs au niveau physique (tolérables) ne se propagent pas par leur nombre aux qubits logiques (irrattrapables).

Hormis l'application de portes, il est possible de mesurer des observables sur les qubits physiques, mesures qui vont servir à corriger les erreurs. Il faut faire attention aux observables que l'on mesure. Comme l'on a vu à la section 2.1.2, si la mesure apporte de l'information sur l'état des qubits, on perd tout ou partie des propriétés de superpositions quantiques.

L'idée des codes stabilisateurs est de *choisir les états des qubits physiques sous-jacents aux qubits logiques de sorte que la mesure de certaines observables* (données par la matrice de parité du code linéaire classique associé au code stabilisateur) *n'apporte aucune information sur l'état des qubits logiques sous peine d'effrondement de leur état quantique.*

Par exemple, supposons que l'on dispose de deux qubits physiques pour encoder un qubit logique. Une façon de le faire pourrait être de choisir comme base physique associée à la base logique (il est d'usage de noter les qubits logiques avec un indice $_L$) :

$$|0\rangle_L = |00\rangle \qquad Z_1 Z_2 |0\rangle_L = Z|0\rangle \otimes Z|0\rangle = |0\rangle \otimes |0\rangle = |0\rangle_L$$
$$|1\rangle_L = |11\rangle \qquad Z_1 Z_2 |1\rangle_L = Z|1\rangle \otimes Z|1\rangle = -|1\rangle \otimes - \otimes |1\rangle = |1\rangle_L$$

Ces deux kets sont propres et associés à la valeur propre 1 pour l'opérateur $Z_1 Z_2$. Pour cette raison, tout état logique quelconque $|\varphi\rangle_L = \alpha |0\rangle_L + \beta |1\rangle_L$ est également invariant par application de $Z_1 Z_2$. Par conséquent, seule la valeur propre 1 peut résulter de la mesure de $Z_1 Z_2$ sur $|\varphi\rangle_L$ s'il n'a subi aucune erreur. Cette mesure n'apporte donc aucune information et n'altère donc pas les proriétés quantiques du système.

Ce code ressemble au code à répétition, à la différence fondamentale que l'on n'encode pas $|\varphi\rangle_L$ par $|\varphi\rangle \otimes |\varphi\rangle$ mais en ajoutant la redondance d'information sur la base (c'est suffisant car les erreurs que l'on souhaite corriger n'altèrent pas le module des coefficients de $|\varphi\rangle$).

Ce code a des performances médiocres. Ce code dispose d'un opérateur logique $X_L = X_1X_2$ (puisque $X_1X_2\,|0\rangle_L = |1\rangle_L$ et $X_1X_2\,|1\rangle_L = |0\rangle_L$). Z_1 ou Z_2 peuvent convenir comme opérateurs logiques Z_L car ils agissent correctement sur la base logique, mais ils sont identiques aux erreurs de type Z qui surviennent sur le qubit physique 1 ou 2, ce qui fait qu'on ne sera pas capable de détecter d'erreurs de type Z. Ce code simple permet néanmoins de détecter une erreur de type X, sans pouvoir la localiser et donc la corriger. En effet, si une erreur applique X_1 par exemple sur $|\varphi\rangle_L$, on obtient :

$$|\widetilde{\varphi}\rangle = X_1\,|\varphi\rangle_L = \alpha\,|10\rangle + \beta\,|01\rangle\,, \quad Z_1Z_2\,|\widetilde{\varphi}\rangle = -\alpha\,|10\rangle - \beta\,|01\rangle = -\,|\widetilde{\varphi}\rangle$$

Au lieu d'être associé à la valeur propre 1 de Z_1Z_2, l'état erroné $|\widetilde{\varphi}\rangle$ est ket propre associé à la valeur propre -1. Ainsi, si l'on mesure Z_1Z_2 sur un état $|\psi\rangle$, on sait que 0 ou deux qubits physiques ont reçu une erreur de type X si l'on obtient 1, et un seul si l'on obtient -1. On est donc capable de détecter une erreur de type X.

En ajoutant un troisième qubit physique, on peut même corriger une erreur de type X. En effet, le code possède dorénavant trois opérateurs pour lesquels un qubit valide est invariant. Ce sont des opérateurs de stabilisation.

$$\begin{aligned} |0\rangle_L &= |000\rangle & Z_1Z_2\,|0\rangle_L &= Z_1Z_3\,|0\rangle_L = Z_2Z_3\,|0\rangle_L = |0\rangle_L \\ |1\rangle_L &= |111\rangle & Z_1Z_2\,|1\rangle_L &= Z_1Z_3\,|1\rangle_L = Z_2Z_3\,|1\rangle_L = |1\rangle_L \end{aligned}$$

On dispose de trois opérateurs de stabilisation par l'application desquels le code est invariant (deux seulement sont indépendants, le troisième étant égal au produit des deux autres). En les mesurant comme précédemment, on est capable pour chaque Z_iZ_j de décider si une erreur de type X est survenue sur l'un des qubits i ou j. En considérant les informations pour les deux couples $(1,2),(1,3)$, on est capable de localiser l'erreur éventuelle (la mesure de $(2,3)$ est inutile car redondante). En effet, si une seule erreur X est survenue sur le qubit k, les mesures de Z_iZ_k et Z_jZ_k vaudront -1, tandis que la mesure de Z_iZ_j vaudra 1. Il suffit alors d'appliquer l'opérateur X sur le qubit physique k pour corriger l'erreur. Ce code est donc meilleur que le précédent, d'autant plus qu'il possède un opérateur logique $X_L = X_1X_2X_3$ et un opérateur logique $Z_L = Z_1$. Il est cependant incapable de détecter des erreurs de type Z (cela s'explique par l'absence d'opérateur de stabilisation contenant un opérateur physique X_i). Il faut donc encore complexifier le code, et donc considérer plus de qubits physiques.

Il est plus commode de changer de point de vue. Au lieu de partir des états physiques et de regarder quels sont les opérateurs de stabilisation associés, il est plus simple de partir d'opérateurs de stabilisation dont on sait qu'ils permettent les corrections d'erreurs souhaitées et d'en déduire les états physiques qui codent la base logique. C'est la démarche des codes CSS.

2.2.3 Codes CSS (Robert Calderbank, Peter Shor et Andrew Steane)

En partant de la matrice de syndrome de certains codes linéaires classiques aux performances connues, on peut trouver un ensemble d'opérateurs de stabilisation permettant de corriger le même nombre d'erreurs de chaque type [1][5]. Par exemple pour le code de Hamming [7, 4, 3] :

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

On en déduit les six opérateurs de stabilisation (pour chaque ligne, on remplace tous les 1 soit par X soit par Z et on remplace les 0 par l'opérateur identité I) :

$$\begin{array}{ll} X_1X_2X_3X_4 & Z_1Z_2Z_3Z_4 \\ X_1X_2X_5X_6 & Z_1Z_2Z_5Z_6 \\ X_1X_3X_5X_7 & Z_1Z_3Z_5Z_7 \end{array}$$

La base logique est alors décrite par les états physiques suivants (il est aisé de vérifier qu'ils sont invariants par application des opérateurs ci-dessus) :

$$\begin{aligned} |0\rangle_L = \quad & \frac{1}{\sqrt{8}}\left(|0000000\rangle + |1111000\rangle + |1100110\rangle + |0011110\rangle + \right. \\ & \left. |1010101\rangle + |0101101\rangle + |0110011\rangle + |1001011\rangle\right) \\ |1\rangle_L = \quad & \frac{1}{\sqrt{8}}\left(|1111111\rangle + |0000111\rangle + |0011001\rangle + |1100001\rangle + \right. \\ & \left. |0101010\rangle + |1010010\rangle + |1001100\rangle + |0110100\rangle\right) \end{aligned}$$

Ce code (CSS[7,1,3]) est un code quantique correcteur d'erreurs, encodant 1 qubit logique sur 7 qubits physiques avec une distance de 3 i.e. capable de corriger 1 erreur de type quelconque parmi les qubits physiques. De plus, une grande partie des opérateurs que l'on voudrait appliquer sur des qubits logiques codés selon ce code $(H, CNOT, S, X, Z)$ sont implémentables en appliquant des portes quantiques qubit physique à qubit physique, ce qui empêche la propagation d'erreur au sein d'un même qubit logique.

2.3 Concaténation de codes quantiques

Si l'on souhaite augmenter la fiabilité d'une machine quantique, on doit diminuer le nombre d'erreurs au niveau des qubits logiques que l'on manipule. Sans code correcteur d'erreurs, la moindre erreur sur un qubit physique ruine le calcul. La probabilité d'échec devient alors la probabilité d'erreur physique. Si l'on utilise un code capable de corriger une erreur, la probabilité d'erreur descend à Cp_e^2 où p_e^2 est environ la probabilité qu'un nombre incorrectible d'erreurs physiques se produise et C une constante positive caractérisant un surcoût dû à la géométrie du code.

Une solution pour réduire la probabilité d'erreurs est la concaténation. Le principe est de coder des qubits logiques avec d'autres qubits logiques au lieu d'utiliser des qubits physiques. Supposons que l'on pratique deux niveaux de concaténation du même code CSS[7, 1, 3]. Pour le niveau logique supérieur, on a besoin de 7 qubits logiques. Pour chacun de ces qubits, on a besoin de 7 qubits physiques. La probabilité d'échec du calcul est donc environ $Cp_e'^2$ où C désigne le surcoût du code CSS et p_e' la probabilité d'erreur sur un qubit logique, c'est-à-dire que la probabilité d'échec du calcul est environ $C^3p_e^4$. Pour un niveau de concaténation k, on utilise 7^k qubits physiques, mais la probabilité d'erreurs est $\dfrac{(Cp_e)^{2^k}}{C}$ [3][1].

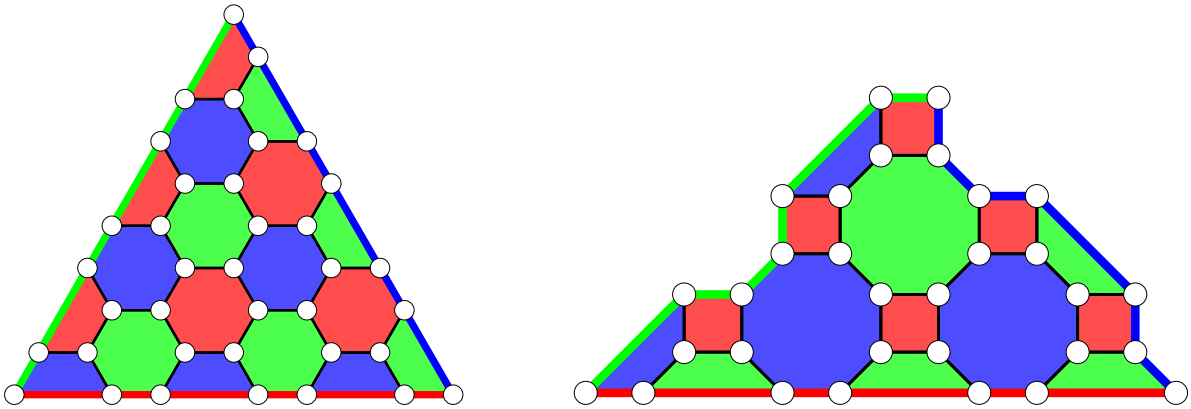
On a mis en évidence le résultat fondamental du calcul quantique tolérant aux fautes. Sous l'hypothèse que la probabilité d'erreurs physiques p_e est inférieure à une valeur seuil $\left(\dfrac{1}{C}\right)$, on peut réaliser des calculs dont le taux d'erreur est aussi petit que l'on veut. Ce stage a pour but de s'intéresser à l'évaluation des performances de la procédure d'encodage et de décodage d'une classe de codes stabilisateurs particulière, les codes couleur en deux dimensions.

2.4 Codes couleur topologiques

La question de l'implémentabilité physique est également cruciale. En effet, un code extrêmement performant mais inextricablement compliqué risque de ne pas être pratiquement réalisable (il faut pouvoir placer les cablâges). Partir de la géométrie finale désirée du code quantique (au niveau des qubits physiques) semble être une bonne idée.

Ce stage s'intéresse en particulier aux codes topologiques en deux dimensions, c'est-à-dire tels qu'il est possible de disposer les qubits physiques sur la frontière (la face extérieure) d'un objet en 3 dimensions (sphère, tore, g-tore...). De façon générale, il est possible d'encoder sur un g-tore (une sphère avec g anses) $2g$ qubits en pavant entièrement la surface de qubits physiques [6]. Nous nous sommes intéressés à un pavage de la sphère moins un sommet pour obtenir un disque (car c'est la forme la plus simple à réaliser). Ce disque est capable d'encoder un qubit logique.

Pour le code couleur, on se dote d'un pavage de la surface d'une sphère, de sorte que chaque sommet soit trivalent (et donc participant à trois faces) et de sorte que les faces obtenues soient 3-coloriables. À cela, on ôte un sommet et les trois faces attenantes.



(a) Qubit logique d'un code couleur hexagonal (b) Qubit logique d'un code couleur 4-8-8

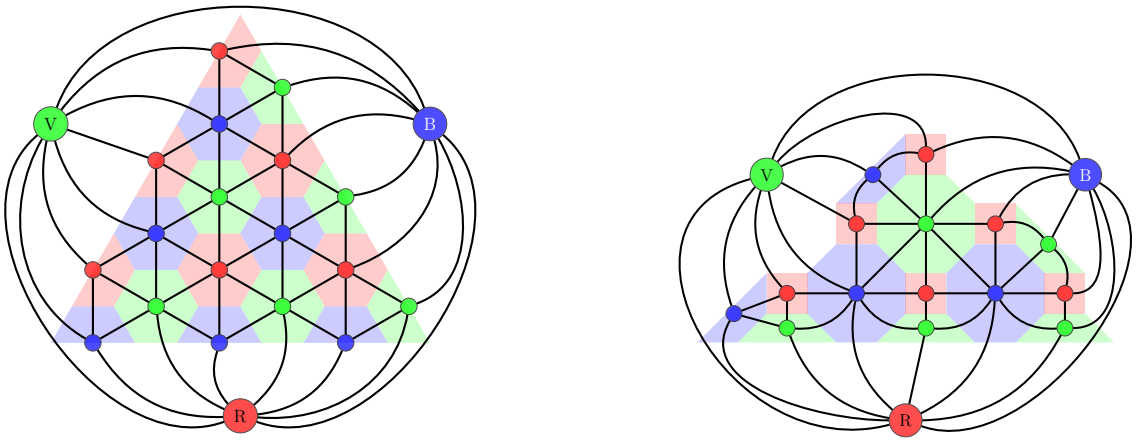
FIGURE 2 – Codes couleurs corrigeant trois erreurs obtenus à partir de différents patterns pavant le plan

On place un qubit à chaque sommet et l'on considère comme opérateurs de stabilisation les opérateurs comportant uniquement l'opérateur X ou l'opérateur Z au niveau des qubits intervenants dans une face colorée donnée et l'opérateur identité ailleurs. Pour le code couleur hexagonal (Fig. 2a), on obtient majoritairement des opérateurs de stabilisation agissant sur six qubits physiques, hormis au bord. Sur la figure 2b, les opérateurs de stabilisation agissent sur quatre ou huit qubits physiques.

On remarque que les faces sur le bord sont tronquées pour former des frontières colorées. Une frontère colorée doit être composée d'une alternance d'arêtes participant à des faces des deux autres couleurs. Cela permet au graphe dual d'être lui aussi 3-coloriable et simple même au niveau des frontières.

Supposons qu'une erreur de type X (resp. Z) ait lieu sur l'un des qubits (l'une des faces de la figure 3a ou 3b). Alors, les opérateurs de stabilisation au niveau des sommets de cette face contenant uniquement des opérateurs Z (resp. X) seront insatisfaits lors de leur mesure (la mesure donnera -1 au lieu de 1). On aura alors localisé l'erreur et on sera capable de la corriger (en réalité, la localisation est plus complexe et nécessite un algorithme détaillé plus loin).

Un code couleur code donc chaque qubit logique au sein d'un triangle de taille maîtrisable et ses opérateurs de stabilisation ont également une taille faible (le fait qu'il soit associé à des faces dans le graphe primal assure que leur mesure n'impliquera que des qubits disposés de façon proche, ce qui est techniquement plus simple à réaliser).



(a) Graphe dual du code couleur hexagonal (b) Graphe dual du code couleur 4-8-8

FIGURE 3 – Graphes duals des codes couleur de la figure 2

2.5 Objectifs

On peut remarquer que chaque qubit dans la représentation duale (Fig. 3) est un triangle aux sommets de couleurs différentes. La représentation duale est elle même inscrite au sein du triangle des sommets frontières. Remplacer chaque face duale par un code couleur plus petit, en identifiant les opérateurs de stabilisation sommets du niveau supérieur aux sommets frontières du niveau inférieur est le sujet de ce stage. Il s'agit en effet d'effectuer quelque chose de comparable à la concaténation de codes stabilisateurs déjà évoquée en 2.3 mais avec l'approche topologique des codes de correction d'erreurs, à savoir utiliser des qubits logiques en lieu et place des qubits physiques (Fig. 4).

Plus précisément, les objectifs de ce stage se déclinent ainsi :

Familiarisation avec les concepts manipulés

Les éléments présentés jusqu'ici sont trop spécifiques pour apparaître dans un quelconque cours de niveau Master. La première étape du stage a donc été de se former sur ces concepts, et ce grâce à des ouvrages spécialisés [3], grâce à certains articles de recherche assez pédagogiques [4], [6] ou bien encore grâce à des thèses qui font généralement l'effort de partir de concepts connus [1], [2].

Construction et implémentation des codes couleurs concaténés

Comme il est encore de nos jours difficile d'implémenter physiquement ces codes quantiques, l'étude repose sur une simulation classique. Malgré le cadre théorique bien défini, il ne nous a pas été possible d'utiliser directement une bibliothèque existante pour construire nos codes. L'architecture de l'implémentation a donc été presque entièrement à penser. De plus, la concaténation peut être réalisée en fonction de certains paramètres (Fig. 4) dont les valeurs admissibles ou intéressantes furent à notre discrétion. Néanmoins, l'implémentation se veut être la moins dépendante de ces valeurs admissibles, afin de pouvoir éventuellement servir pour d'autres travaux (utilisant potentiellement des codes couleur auxquels nous ne nous serions pas intéressés par exemple).

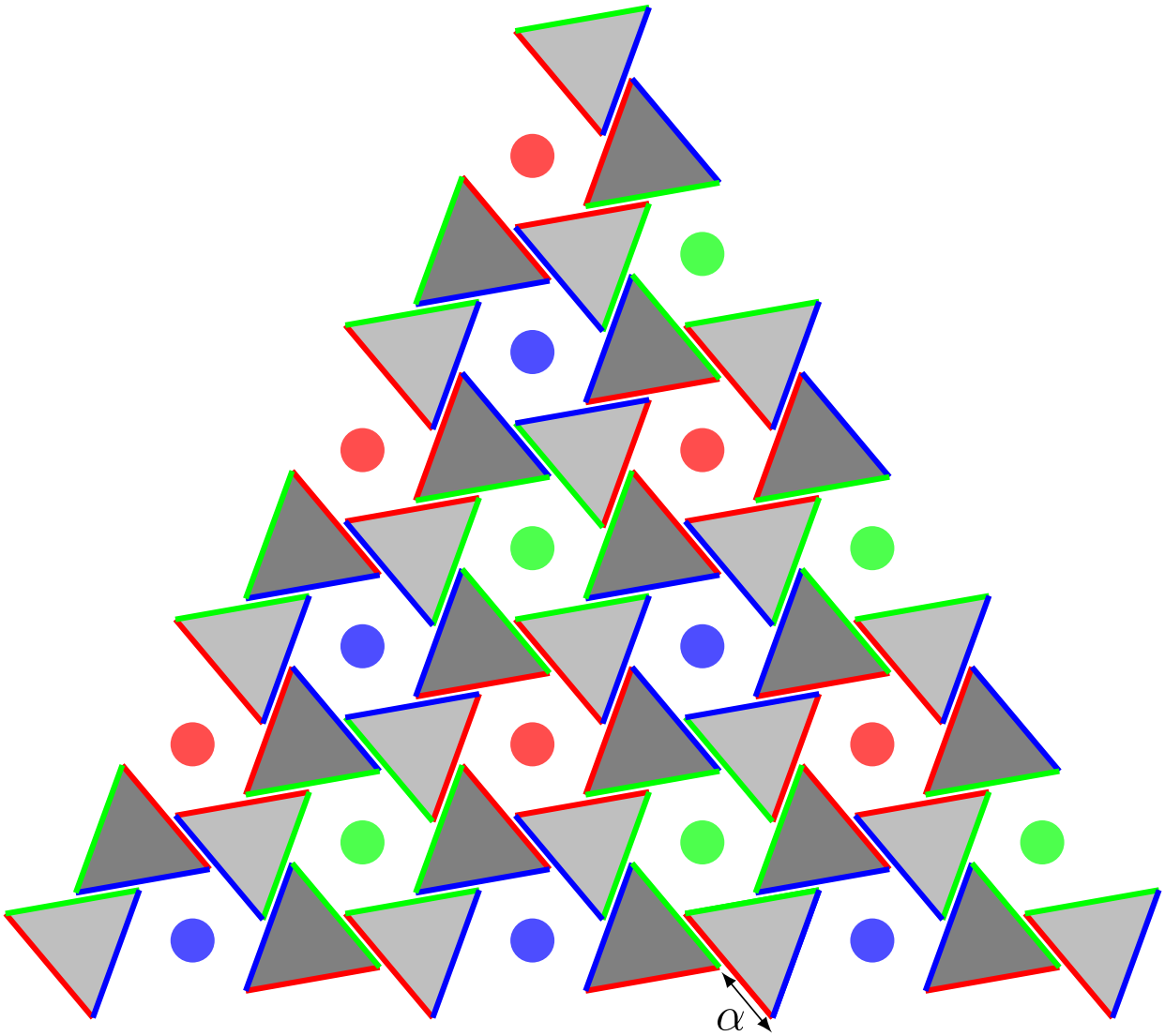


FIGURE 4 – Quasi-concaténation du code couleur hexagonal, avec un décalage $\alpha = 0.5$. Les différentes parties grisées sont analogues à ce qui apparaît à la figure 2a. Au niveau des zones de jonction, les faces des codes de couleurs sont modifiées procéduralement afin de fusionner les codes triangulaires en un seul grand code.

Implémentation du processus de mesures et de décodage

Le processus de décodage et de correction d'erreurs des codes couleur, jusqu'alors succintement évoqué, comporte quelques subtilités et requiert donc un algorithme *ad hoc* que [7] justifie. En plus de l'implémentation de cet algorithme dans notre architecture, une adaptation est nécessaire pour qu'il soit applicable aux codes concaténés. Certains outils déjà développés [8] permettent de simplifier cette implémentation et ont donc contraint notre architecture pour être compatible.

Raffinages du modèle d'erreurs et estimation des performances

Les performances du décodage dépendent des hypothèses sur les stabilisateurs que l'on s'autorise à mesurer sans erreurs. Ces performances sont évaluées statistiquement à partir des simulations effectuées et comparées aux résultats d'autres codes, publiés par exemple dans [9].

3 Solutions étudiées

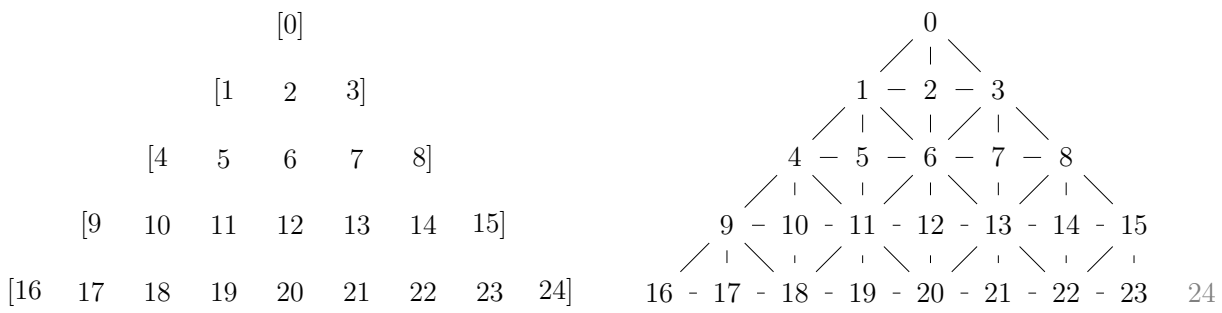
3.1 Construction du code couleur concaténé

N'ayant accès à aucun code *open source* permettant la simulation d'un code couleur, la première étape d'implémentation de ce stage a donc été le codage d'une classe décrivant un code couleur en deux dimensions (`ColorCode2D`). Ayant en tête l'utilisation de la bibliothèque Python `pymatching` pour l'implémentation de l'algorithme de correction d'erreurs, le choix s'est assez naturellement tourné vers Python et sa bibliothèque NetworkX pour représenter des objets tels que les graphes de la figure 3.

Un code couleur (ou son dual en réalité que l'on nommera dorénavant code couleur puisque le primal n'est jamais manipulé) étant un graphe planaire, il m'a semblé naturel de surcharger la classe `PlanarEmbedding` de NetworkX pour implémenter les codes couleur, d'autant plus qu'un code couleur concaténé est également visible comme un code couleur. Des constructeurs particuliers permettent de générer les codes hexagonaux (Fig. 3a) et 4-8-8 (Fig. 3b). En plus de leur label et de leur couleur, un dictionnaire garde trace du niveau de concaténation d'où provient chaque sommet (opérateur de stabilisation) signifiant en particulier s'il s'agit d'un sommet frontière ou interne. Un point délicat est l'obligation de donner les voisins de chaque sommet dans le sens horaire.

Pour générer un code couleur 4-8-8 corrigeant n erreurs, on commence par lister les $(n+2)^2$ premiers entiers. On découpe cette liste en listes de longueur impaire et de taille croissante (Fig. 5a). On décide pour chaque entier s'il sera tétravalent ou octovalent (s'il est d'indice impair dans sa liste dans la vue pyramidale ou non) et on l'associe à une liste vide de voisins. Par exemple, 6 est octovalent, tout comme 4 ; 2 est tétravalent.

Pour chaque entier, on ajoute à sa liste de voisins les entiers immédiatement au-dessus (trois ou un suivant qu'il ait été choisi octovalent ou tétravalent). Par exemple, les voisins de six deviennent [1, 2, 3]. Puis, pour chaque entier, on ajoute à sa liste de voisins l'entier directement à droite (6 : [1, 2, 3, 7]). Puis, pour chaque entier, on ajoute à sa liste de voisins les entiers immédiatement au-dessous, de droite à gauche (trois ou un suivant qu'il ait été choisi octovalent ou tétravalent) (6 : [1, 2, 3, 7, 13, 12, 11]). Enfin, pour chaque entier, on place dans sa liste de voisins l'entier immédiatement à gauche, en prenant garde aux bords (Fig. 5b). On a bien [1, 2, 3, 7, 13, 12, 11, 5] comme voisins de 6.

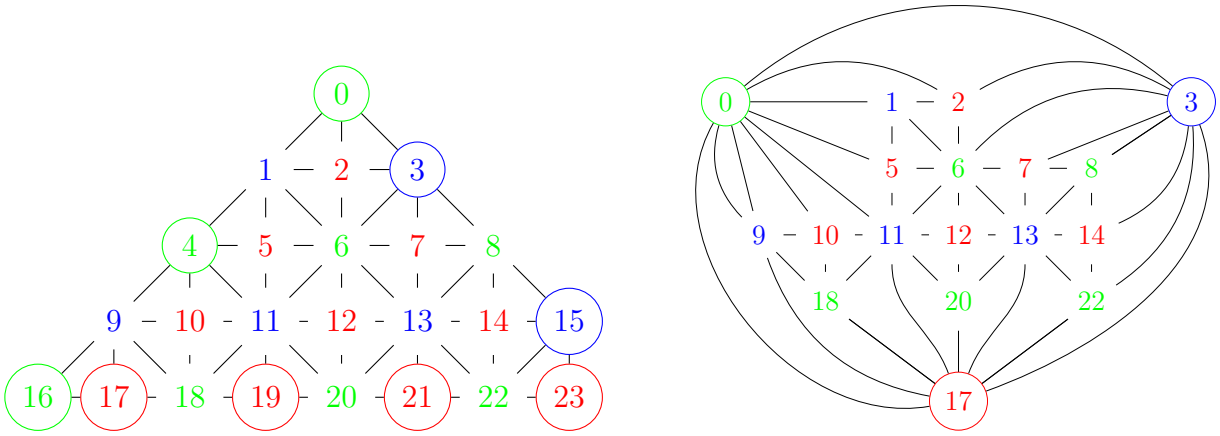


(a) Étape 1 (b) Étape 2

FIGURE 5 – Procédure de création d'un code couleur 4-8-8 corrigeant trois erreurs (1)

À ce moment, on attribue une couleur à chaque entier. Les entiers choisis tétravalents sont d'une couleur (ici, rouge). Tous les entiers octovalents d'une strate seront de la même couleur et on alterne vert et bleu d'une strate à l'autre. On choisit également aux extrémités des strates des entiers frontières tels 3, 4 ou 19 (Fig. 6a).

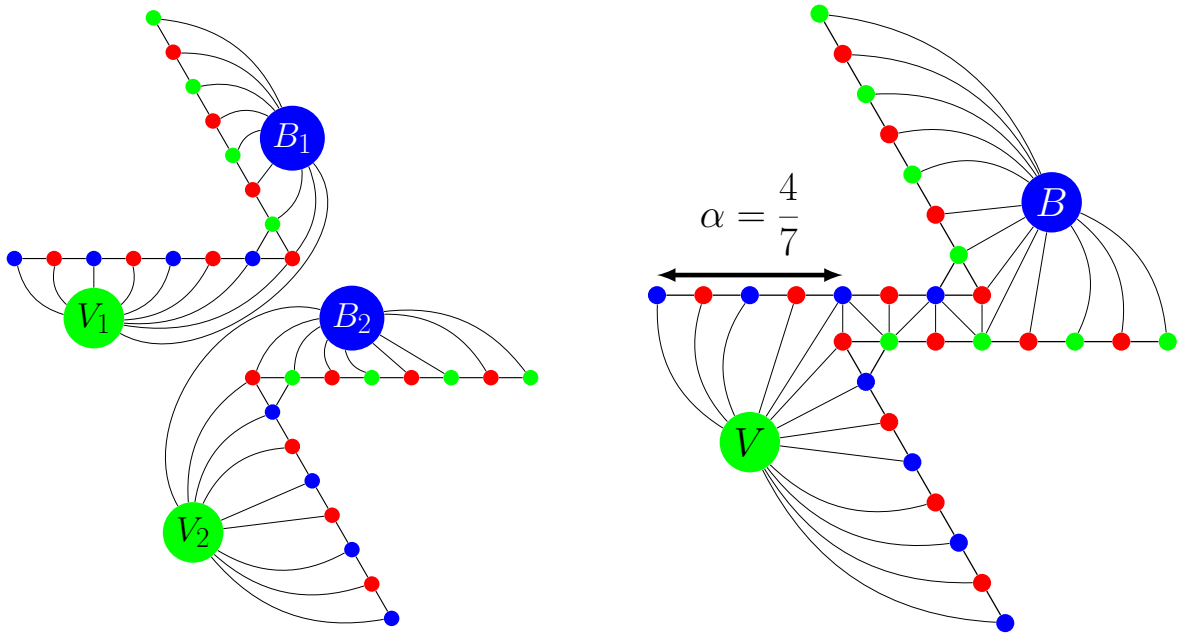
L'ultime étape consiste à fusionner les listes de voisins de chaque entier frontière de la même couleur et de n'en conserver qu'un. On peut aisément voir que les entiers restants correspondent exactement aux sommets de la figure 3b, légèrement réagencés.



(a) Étape 3 (b) Étape 4

FIGURE 6 – Procédure de création d'un code couleur 4-8-8 corrigeant trois erreurs (2)

Ensuite, il faut une procédure pour fusionner deux codes couleur adjacents. J'ai donc écrit une fonction prenant en entrée deux codes couleur (Fig. 7a), l'indication des frontières à fusionner ainsi que le décalage à opérer et retournant le code couleur résultant de la fusion des deux précédents. Cette procédure fait nécessairement apparaître des sommets hexavalents, sans pour autant que l'on augmente le nombre de qubits (faces) ou de sommets non frontières (opérateur de stabilisation) (Fig. 7b).

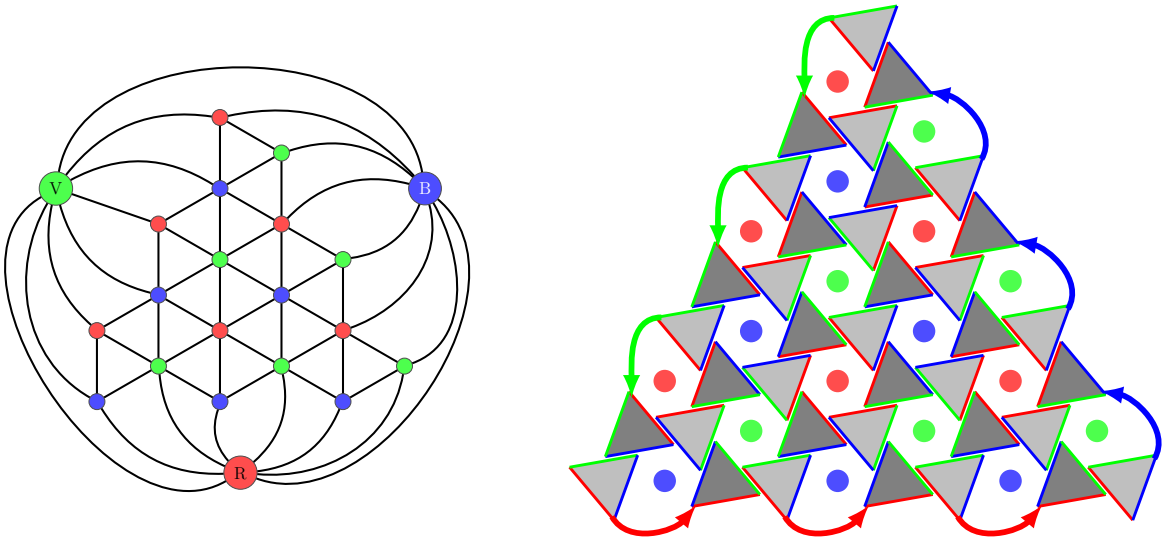


(a) Configuration initiale (b) Configuration finale

FIGURE 7 – Procédure de fusion de deux codes couleur (partiellement dessinés)

La dernière étape pour la concaténation de codes couleur est l'automatisation des appels nécessaires à la fonction précédente. Pour chacune des 37 faces triangulaires du code couleur de niveau supérieur (Fig. 8a), on instancie une copie aux labels uniques (par *deepcopy* relabelisée du code couleur de niveau inférieur).

Ensuite, pour chacune des 54 arêtes entre ces faces, on fusionne les sommets frontières des deux codes de niveau inférieur. Pour choisir la couleur des frontières, on prend garde à utiliser les mêmes couleurs que les sommets de l'arête considérée. À la fin, on obtient une figure analogue à la figure 8b. Les flèches désignent les fusions au niveau des sommets tétravalents ne pouvant être représentées de manière esthétique.



(a) Niveau supérieur (b) Code concaténé (un niveau)

FIGURE 8 – Concaténation de codes couleur hexagonaux

3.2 Implémentation du décodeur de codes couleur triangulaires

Afin d'étudier les performances des codes couleur concaténés, il faut d'abord implémenter un décodeur permettant de corriger les erreurs des codes couleur triangulaires. Un décodeur pour les codes couleur a été proposé par [7]. Celui-ci ne permet pas de corriger les codes disposant de frontières, c'est pourquoi c'est plutôt l'algorithme présenté en [10] qui a d'abord été implémenté. Son objectif est simple : pour un syndrome mesuré donné causé par certaines erreurs, proposer un ensemble de qubits à inverser corrigeant le syndrome et ayant le maximum de chances de corriger les erreurs physiques sans introduire d'erreurs logiques.

Sur la figure 9, on voit certains qubits erronés (cercles grisés) induisant un syndrome regroupant les opérateurs de stabilisation insatisfaits (cercles colorés entourés). Un opérateur de stabilisation est dit insatisfait s'il est adjacent à un nombre impair de qubits erronés.

L'algorithme de décodage n'a connaissance que des opérateurs insatisfaits pour proposer la correction. Il retourne un ensemble de qubits dont l'inversion permet la correction systématique du syndrome, i.e. tel que tous les opérateurs de stabilisation soient adjacents à un nombre pair de qubits de la différence symétrique de l'erreur et de la correction. Cette correction peut s'effectuer de différentes manières, introduisant ou non une erreur logique. Une correction correcte (illustrée en violet à la figure 10) corrige l'erreur aux opérateurs de stabilisation près (si tous les qubits adjacents au même opérateur de stabilisation sont inversés, l'erreur est tout aussi bien corrigée que si aucun d'entre eux ne l'était). À l'inverse, la correction en orange (Fig. 10) induit une erreur logique, car il demeure une chaîne de qubits inversés reliant les trois frontières à l'issue

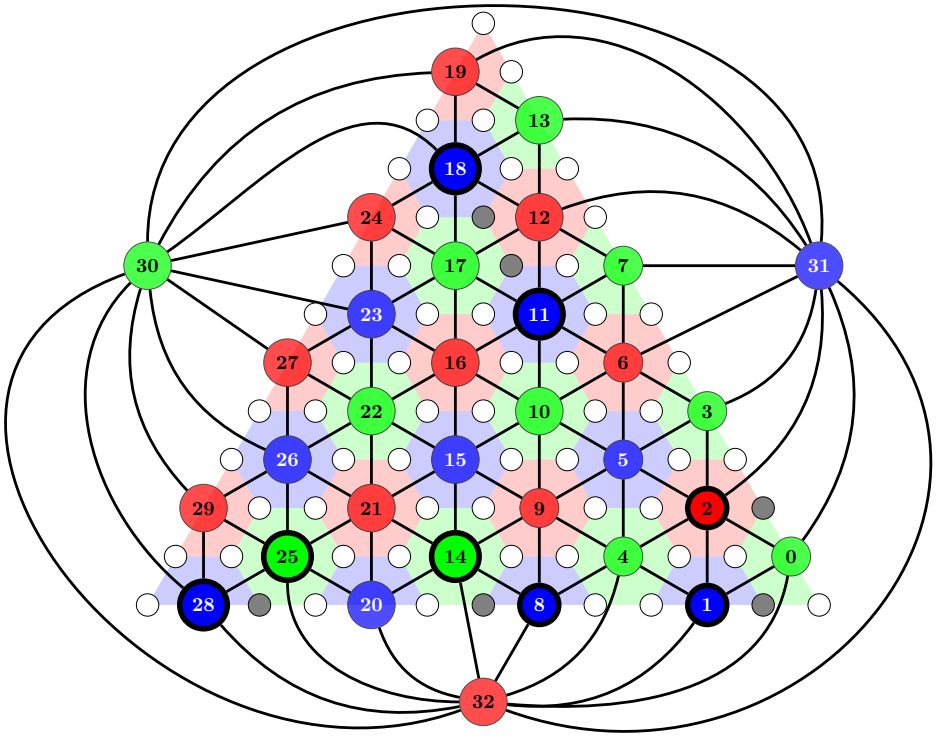


FIGURE 9 – Erreurs physiques (gris) et syndrome mesuré associé (sommets opaques)

de la correction.

Un bon décodeur doit être capable de corriger un nombre maximal d'erreurs (idéalement égal au nombre d'erreurs correctibles théorique). De plus, pour que le théorème mis en évidence en 2.3 s'applique, il faut également que le décodeur puisse décoder une part importante des erreurs de poids linéaire en la taille du code (et donc dépassant la distance théorique), c'est pourquoi la figure 9 et les suivantes comportent souvent un nombre d'erreurs supérieur au nombre d'erreurs théoriquement correctibles. Ces deux critères servent à évaluer les performances du décodeur implémenté.

Les codes couleur sont similaires à une autre classe de codes (les codes de surface) pour lesquels il existe un décodeur efficace. L'idée est donc de se ramener à des problèmes de décodage de codes de surface, de les décoder avec la bibliothèque `pymatching`, puis de synthétiser les résultats pour obtenir la correction pour le code couleur.

Il est important de remarquer que chaque qubit (cercle blanc/gris) est entouré par un triangle comportant un sommet de chaque couleur. Ainsi, si l'on fait abstraction de tous les sommets d'une couleur donnée (et de toutes les arêtes incidentes à ces sommets) dans un code couleur, il est toujours possible d'associer de façon unique chaque qubit à une arête restante (hormis l'arête extérieure). On obtient alors un sous-graphe restreint semblable à la définition des codes de surface. Le syndrome se retrouve lui aussi amputé des sommets de la couleur enlevée. On applique alors le décodeur pour les codes de surfaces et l'on récupère un appariement des sommets de ce syndrome restreint via des arêtes que l'on colore de la couleur enlevée [7]. On répète ce processus pour les trois couleurs possibles (Fig. 11, 12).

Néanmoins, la présence des trois frontières rend les choses plus complexes, et c'est l'objet de [10] de proposer une marche à suivre dans ce cas. Cet article comporte certaines zones d'ombre et est parfois peu clair sur la façon dont certains choix sont faits. J'ai donc du prendre certaines

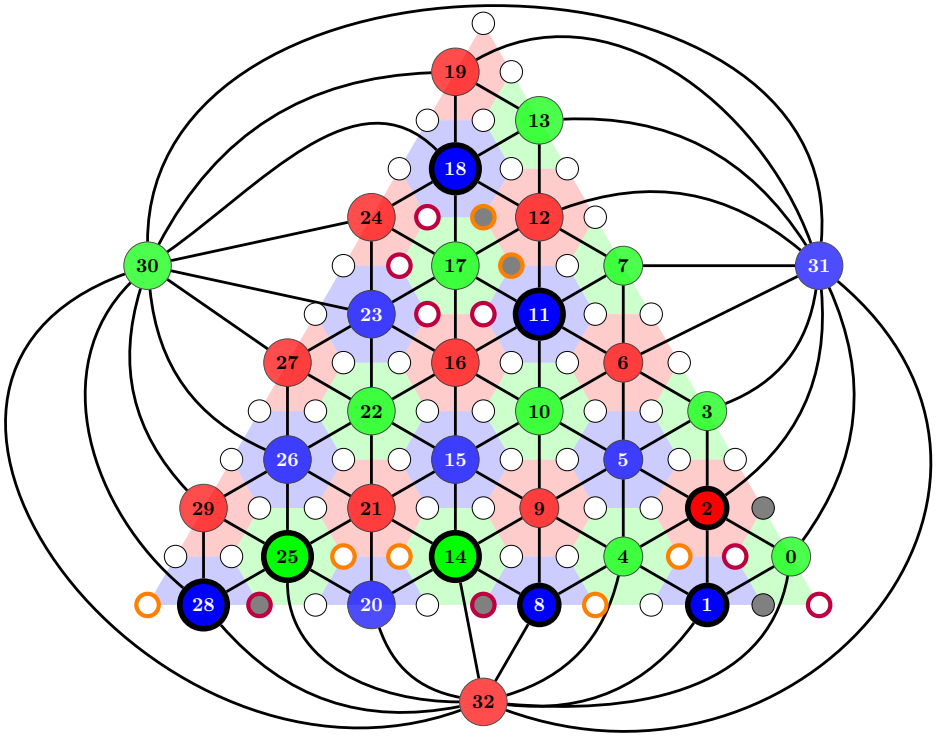


FIGURE 10 – Deux corrections possibles, logiquement erronée ou correcte

initiatives et de la distance avec ce dernier.

Le décodage des codes de surface repose sur un appariement deux à deux des sommets du syndrome. Ceci n'est possible que si le syndrome est de cardinal pair. Cependant, rien n'indique que les syndromes restreints aient cette propriété. Heureusement, on a la possibilité d'ajouter l'une des deux frontières du graphe restreint au syndrome pour garantir un cardinal pair. Il y a un choix à faire parmi les deux frontières candidates et le plus simple est de calculer les deux appariements et de conserver celui de poids minimum. Cet ajout est légitime car les sommets frontières ne sont pas des opérateurs de stabilisation et peuvent ici servir de degrés de liberté.

De façon plus concrète, il m'a fallu rendre possible la génération des différents sous-graphes restreints d'un `ColorCode2D` en les rendant compatibles avec les exigences de la classe `Matching` de `pymatching` sur laquelle le décodeur des codes de surface doit être appelé.

Une fois les trois appariements minimaux calculés (Fig. 11), il faut encore décider d'une correction. Bien que [10] utilise une notion peu claire de *connected components*, j'ai pu reformuler l'algorithme pour qu'il travaille sur des cycles, tout en donnant des corrections équivalentes à la formulation initiale. Comme le graphe des appariements obtenus comporte uniquement des sommets ayant un nombre pair d'arêtes incidentes (il peut être nécessaire d'ajouter une arête reliant deux frontières pour que cela soit le cas), il est possible de partitionner les arêtes de ce graphe en cycles disjoints. Il y a alors trois types de cycles identifiables. Tout d'abord, les cycles de type 0 ne traversant aucun sommet frontière. Puis, les cycles de type 1 qui passent par au plus deux sommets frontières. Enfin, les cycles de type 2 qui passent par les trois sommets frontières.

L'étape suivante est une étape de marquage. Au sein de chaque cycle, on marque certains sommets avec le label de ses voisins au sein du cycle. Le choix des sommets marqués dépend du type de cycle. Pour un cycle de type 0, il faut choisir tous les sommets d'une des trois couleurs

sans préférence. Pour les cycles de type 1, il faut choisir tous les sommets qui ne sont pas d'une couleur identique à la couleur d'un sommet frontière traversé par le cycle. Pour les cycles de type 2, une étape supplémentaire est nécessaire : il faut séparer chaque cycle en trois chemins reliant les sommets frontières. Au sein de chacun des trois chemins, il est alors possible de marquer tous les sommets qui ne sont pas de la couleur présente aux extrémités du chemin. Tous ces choix de sommets marqués sont faits de sorte à ne jamais marquer un sommet frontière. Un simple dictionnaire permet d'associer chaque sommet marqué à une liste de voisins au sein des cycles.

La dernière étape consiste à choisir pour chaque sommet marqué une des deux moitiés des qubits adjacents. En effet, chaque sommet possède un nombre pair de marques et il est possible de séparer les qubits adjacents à chaque sommets marqués en deux groupes s'alternant aux niveaux des arêtes marquées (Fig. 11). Les qubits d'un des deux groupes choisi arbitrairement seront les qubits à inverser pour appliquer la correction.

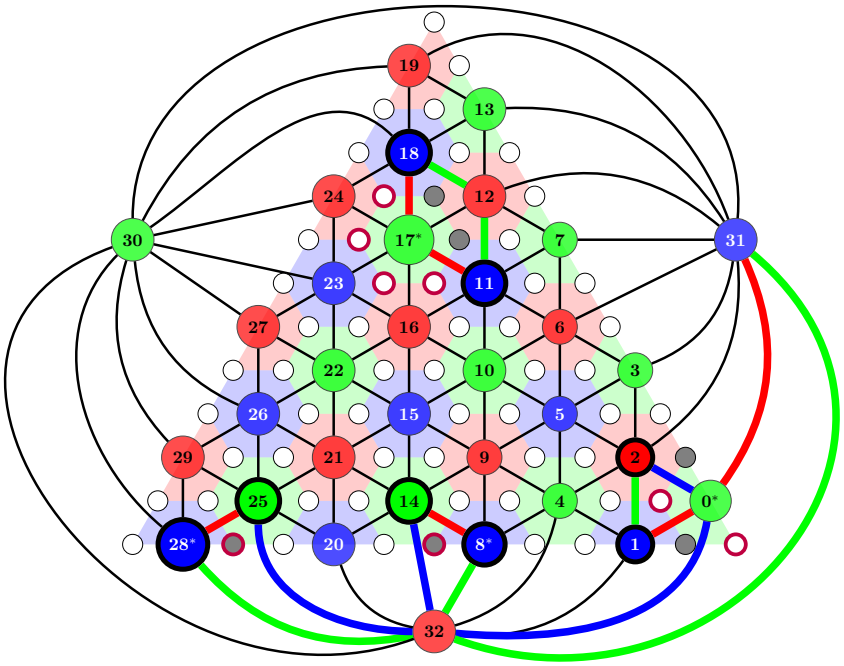


FIGURE 11 – Exemple d'exécution amenant à un décodage correct. Après avoir calculé les trois appariements colorés, on identifie les cycles. Un découpage possible est le suivant : (11-12-18-17) : type 0, (25-28-32-8-14-32) : type 1, (0-1-2-0-31-32-0) : type 1. Les sommets marqués au sein de chaque cycle sont indiqués avec un astérisque. Dans le premier cycle, on aurait pu choisr de marquer 11 et 18, ou 12 au lieu de 17. La présence de 32 dans le second cycle interdit de marquer les sommets rouges, mais le choix des sommets verts aurait pu être fait. Pour le dernier cycle, le seul choix possible est la couleur verte. Au niveau de chaque sommet marqué, les arêtes colorées divisent les qubits adjacents en un nombre pair de groupes et on choisit d'appliquer la correction à la moitié de ces groupes (peu importe laquelle). Au lieu des quatre sommets choisis proche de 17, on aurait pu en choisir deux autres, obtenant la même correction à l'opérateur de stabilisation 17 près. Il y a quatre régions autour de 0, il faut donc en choisir deux qui doivent nécessairement s'alterner.

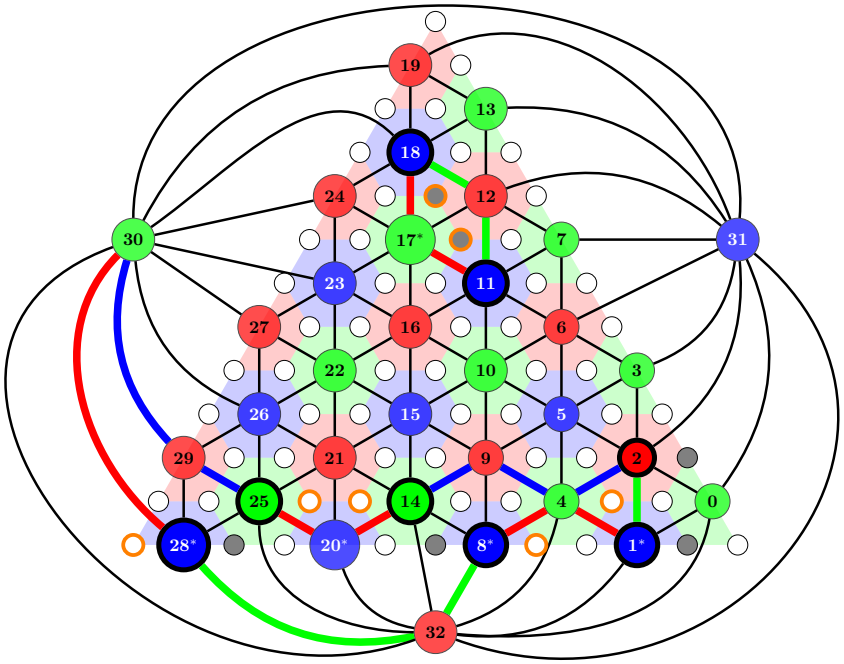


FIGURE 12 – Exemple d'exécution amenant à un décodage incorrect. Un œil averti remarquera que le poids des appariements (le nombre d'arêtes colorées) est légèrement supérieur à celui de la figure 11, ce qui rend l'exemple un peu artificiel. L'algorithme trouve néanmoins une correction au syndrome mesuré induisant une erreur logique. Il existe des exemples où l'algorithme est pris en défaut malgré le choix de l'appariement de poids minimum.

3.3 Évaluation des performances du décodeur

3.3.1 Nombre d'erreurs correctibles

Un décodeur corrige m erreurs survenant au sein d'un code s'il parvient à apporter une correction correcte à toutes les configurations comportant au plus m qubits erronés. Idéalement, $m = t$, nombre d'erreurs théoriques correctibles par le code. En l'absence de preuve du nombre d'erreurs corrigées pour le décodeur de code triangulaire avec frontières, il faut se reposer sur la simulation pour tenter de l'estimer. Un test exhaustif des erreurs est à proscrire car le nombre de configurations croît trop rapidement. La meilleure alternative consiste à chercher parmi les erreurs les plus difficiles à corriger (Fig. 13) et de regarder le poids minimal de l'erreur de ce type non corrigée. Pour chaque qubit, on tente de corriger une ligne de qubits erronés le reliant à chaque frontière séparément. On a donc uniquement $3n$ appels au décodeur à faire. Pour toutes ces configurations, on cherche celle de poids minimum qui induit une mauvaise correction et on regarde l'évolution de ce poids minimum avec l'augmentation de la taille du code. Les résultats présentés à la figure 14 laissent supposer qu'asymptotiquement, le décodeur est capable de corriger toutes les erreurs de poids inférieur aux deux tiers du poids théorique. Ce comportement est cohérent avec les attentes de [10]. Il est important de noter que ces résultats ont été obtenus avec des codes hexagonaux. Les performances pour des codes 4-8-8 sont légèrement moins bonnes, avec une asymptote allant vers les poids correspondant à la moitié du poids théorique.

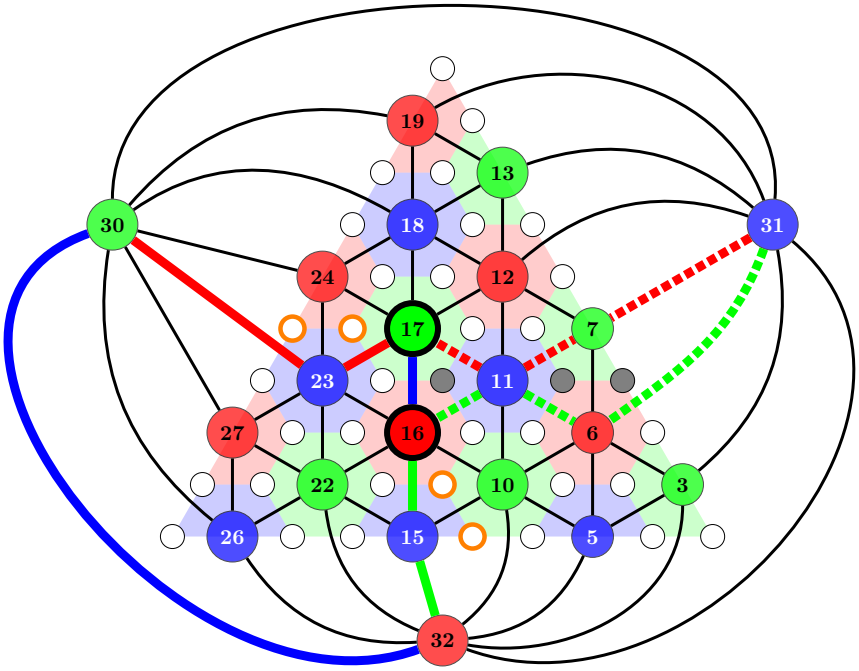


FIGURE 13 – Configuration d'erreur la plus difficile à corriger : les qubits erronés s'alignent vers le centre du code. L'appariement choisi a un poids total de six (traits pleins colorés) mais conduit à la correction erronée orange. L'appariement de poids légèrement plus élevé (en pointillés en plus de l'arête (16-17)) conduirait à la bonne correction.

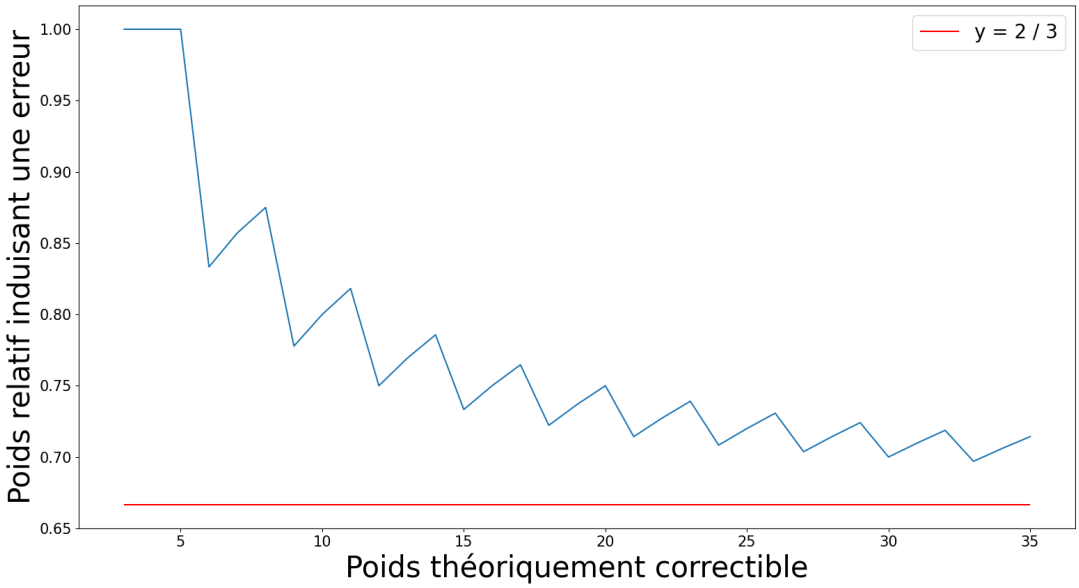


FIGURE 14 – Poids minimal de l'erreur mal corrigée parmi celles qui correspondent à des qubits erronés alignés en fonction du nombre d'erreurs théoriquement correctible du code. L'ordonnée sans dimension correspond au rapport du poids minimal trouvé avec le poids théorique.

3.3.2 Performances contre les erreurs de poids linéaire

Un modèle d'erreur couramment employé consiste à attribuer une erreur à chaque qubit avec une probabilité p_e. On cherche à estimer la probabilité seuil décrite à la section 2.3, c'est-

à-dire la probabilité d'erreur physique maximale pour laquelle on est susceptible d'observer une diminution de la probabilité d'erreur logique en concaténant le code. Pour cela, on applique une méthode de Monte-Carlo : pour différentes probabilités d'erreurs physiques p_e, on tire un grand nombre de configurations d'erreurs (10^4) et on regarde la proportion d'erreurs logiques obtenue après décodage. Cette estimation est reproduite pour des codes de différentes tailles. Les résultats pour des codes hexagonaux sont représentés à la figure 15. On remarque principalement deux choses. D'une part, les courbes se croisent toutes aux alentours de l'abscisse $p_e \approx 0.126$, en accord avec les résultats de [10]. Il s'agit de la probabilité seuil. D'autre part, en deçà de la valeur seuil, on voit que la probabilité d'erreur logique diminue d'autant plus que le code considéré est grand. Ces deux points amènent à penser que le décodeur se comporte convenablement et peut servir de base pour l'étude des codes couleur concaténés.

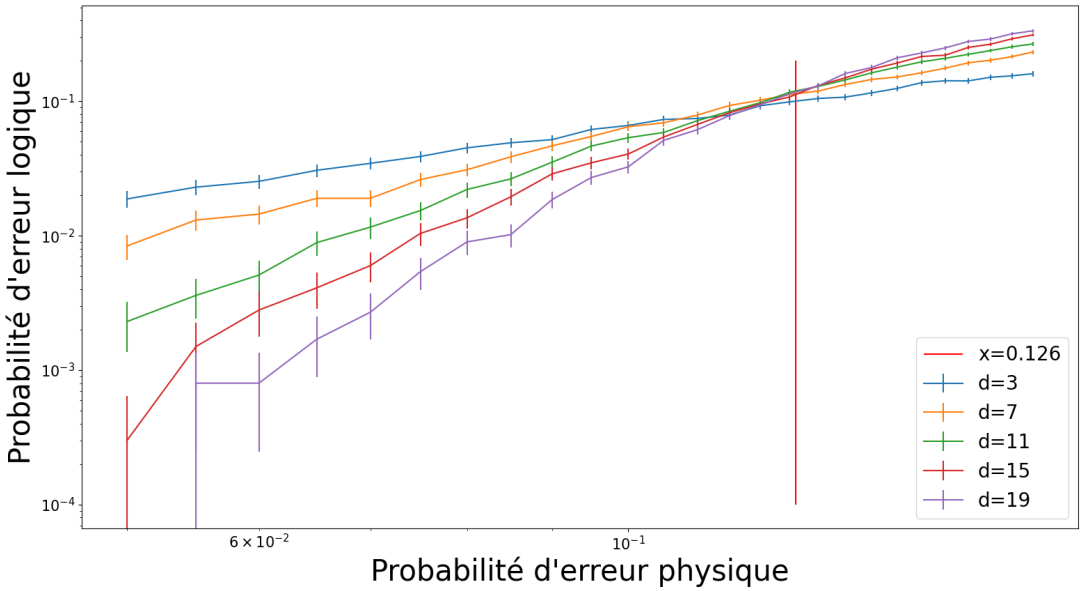


FIGURE 15 – Estimation de la probabilité d'erreur logique pour différents codes hexagonaux et pour différentes probabilités d'erreur physique

3.4 Décoder le code couleur concaténé

3.4.1 Application naïve du décodeur

Les codes couleur concaténés étant des codes couleur, le décodeur est directement utilisable sur ceux-ci. Les résultats obtenus seront très optimistes par rapport à la réalité car l'hypothèse faite implicitement est la possibilité de mesurer les grands opérateurs de stabilisation, opérations qui va introduire des erreurs supplémentaires dans la pratique et donc réduire les performances. Ils donnent néanmoins un aperçu de l'allure des résultats que l'on pourrait obtenir au mieux.

Contrairement à nos attentes, le taux d'erreur logique croît avec la valeur de α. Ces résultats n'enterrent pas nécessairement les codes couleur concaténés mais suggèrent fortement que le décodeur n'est pas pleinement adapté à la structure du code. Certains indices avaient déjà instillé cette idée : le seuil observé lorsqu'on applique le protocole de la figure 15 aux codes couleur 4-8-8 est moindre : le décodeur se comporterait donc plus mal en présence d'opérateurs de stabilisation plus larges, probablement en raison de la multiplication des cas tels que celui illustré à la figure 13. Il s'agit d'un problème, puisque la circonférence de certains opérateurs de stabilisation des codes couleur concaténés est proportionelle à α. Ce décodeur n'est donc pas

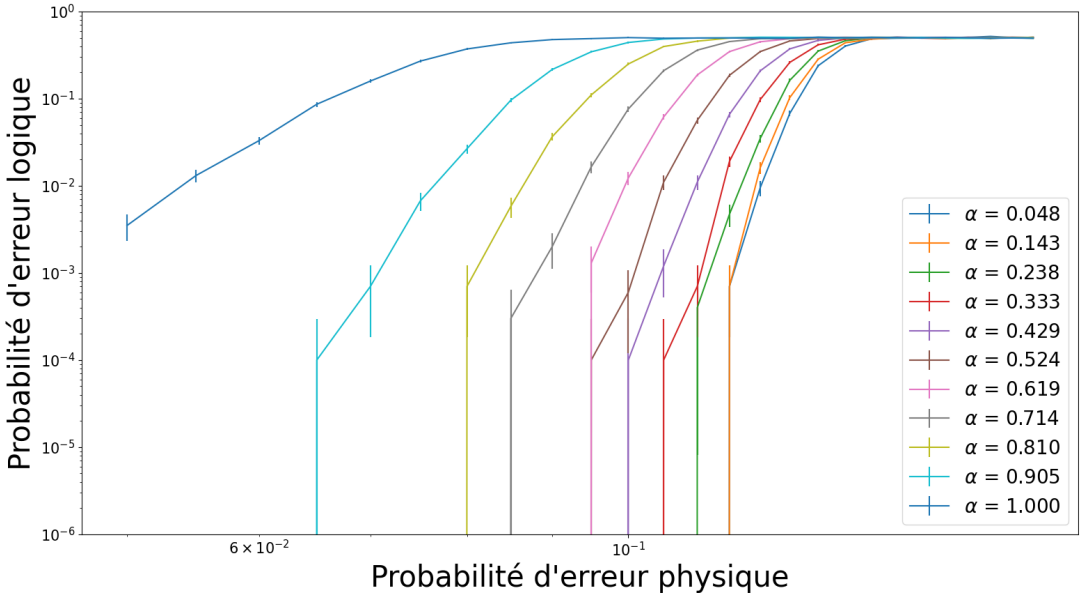


FIGURE 16 – Estimation de la probabilité d'erreur logique pour un code couleur hexagonal corrigeant dix erreurs concaténé avec lui même avec différentes valeurs de α.

utilisable en l'état pour mettre en évidence un éventuel gain dans l'utilisation des codes couleur concaténés. Un décodeur plus complexe ou utilisant d'autres principes semble nécessaire.

3.4.2 Recherche d'autres décodeurs utilisables

D'autres pistes de décodage des codes couleur ont vu le jour [11][12]. À première vue, leur utilisation sur des codes de grandes tailles (donc sur les codes couleur concaténés) aurait un coût exorbitant et c'est pourquoi nous nous sommes concentrés sur une amélioration du décodeur déjà implémenté. Être capable de décoder toutes les erreurs théoriquement correctibles résoudrait tous les problèmes et c'est l'objectif que nous nous sommes initialement fixé. En calculant les quatre appariements de frontières minimaux au lieu d'un seul, nous espérions récupérer parmi eux celui permettant de corriger l'erreur. Néanmoins, il demeure difficile de trouver une procédure permettant de l'identifier et donc d'accéder au décodage optimal. Après quelques semaines infructueuses, une autre piste est explorée : un décodeur hiérarchique.

3.4.3 Décodage hiérarchique

Le décodeur hiérarchique vise à exploiter la nature récursive du code. L'idée est de décoder les qubits logiques intermédiaires du niveau le plus bas vers le niveau le plus haut. Pour cela, on utilise le décodeur décrit en 3.2 sur chacun des qubits logiques intermédiaires (là où il se comporte relativement bien) et on applique les corrections obtenues. Certains changements sont nécessaires sur ces corrections pour qu'elles aient du sens dans le contexte du code concaténé. En effet, le décodeur exploite les degrés de liberté offerts par les sommets frontières, qui n'en sont plus vraiment dans le contexte du code concaténé : si une correction intermédiaire intervient au voisinage d'une frontière, il faut mettre à jour en conséquence le syndrome qui sera utilisé lors de la correction du qubit logique de niveau supérieur. Par ailleurs, les fusions opérées lors de la concaténation ont supprimé les liens entre certains opérateurs de stabilisation et les sommets frontières. En conséquence, il est indispensable de déformer (i.e. appliquer des opérateurs de stabilisation) les cycles passant par les sommets frontières tels qu'illustrés à la figure 11 de sorte qu'ils empruntent bel et bien des arêtes qui existent dans le contexte du code

concaténé. Au niveau supérieur, le code est également un code couleur classique donc corrigé de manière appréciable. Les corrections appliquées pour cette étape requièrent l'inversion des qubits logiques, qui est simplement faite en inversant chacun des qubits physiques du qubit logique.

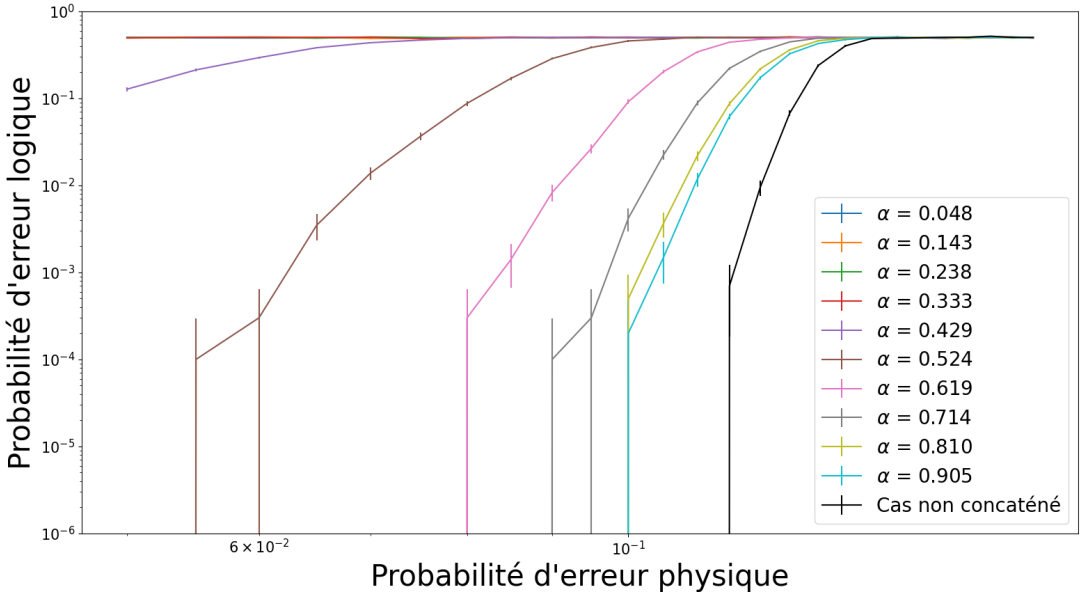


FIGURE 17 – Estimation de la probabilité d'erreur logique imputable au décodeur hiérarchique pour un code couleur hexagonal corrigeant dix erreurs, concaténé avec lui même pour différentes valeurs de α.

Contrairement au décodage précédent, on retrouve le comportement que l'on recherchait avec ce nouveau décodeur : on obtient des performances honorables et d'autant meilleures qu'α grandit (Fig. 17). Cependant, on peut voir en noir que le code non concaténé décodé par le premier décodeur dispose de meilleures performances. Ainsi, le décodage hiérarchique permet d'améliorer le décodage en étant moins sensible à la présence des gros opérateurs de stabilisation. Cependant, ce décodeur hiérarchique n'est pas assez performant pour mettre en évidence un avantage des codes couleur concaténés.

Une des hypothèses faites par le décodeur n'est plus applicable ici : la probabilité d'erreur pour les qubits logiques intermédiaires n'est plus la même partout mais dépend des chances de réussite de chaque décodage au plus bas niveau. Il devrait être possible d'estimer ces chances de réussite à partir de la taille des syndromes observés lors de chacun des décodages.

3.5 Prise en compte des probabilités d'erreur physique différentes

Le calcul des appariements des opérateurs de stabilisation insatisfaits du syndrome (*cf.* 3.2) est fait au sein d'un graphe où les arêtes ont toutes un poids de 1. Lorsque deux opérateurs sont appariés, on considère que chaque arête de l'appariement est adjacente à exactement un qubit erroné et un qubit correct. Au lieu d'un poids de 1, on peut mettre un poids de $-\log(\frac{p}{1-p})$ où p est la probabilité qu'exactement un qubit sur les deux adjacents à l'arête soit erroné. Si tous les qubits ont une probabilité d'erreur physique p_e, on a $p = 2p_e(1-p_e)$ et toutes les arêtes ont le même poids. Le poids d'un appariement de n arêtes est donc $-\log(\frac{p^n}{(1-p)^n})$. En prenant l'opposé et en passant à l'exponentielle, puis en multipliant par $(1-p)^{2n}$, on retrouve la probabilité liée à l'appariement. Le log permet de transcrire la sommation des poids des arêtes

d'un graphe en multiplication des probabilités des événements indépendants associés à chaque arête, permettant ainsi de choisir l'appariement avec le maximum de vraisemblance.

Si maintenant une arête i se trouve entre un qubit de probabilité d'erreur p_1 et un autre de probabilité d'erreur p_2, alors la probabilité qu'exactement l'un des deux soit erroné est $p_i = p_1(1-p_2) + p_2(1-p_1)$ et un poids $-\log(\frac{p_i}{1-p_i})$ est associé à l'arête.

La figure 18 compare les performances du décodeur avec et sans ce changement de poids des arêtes : on tire au hasard environ un tiers des qubits qui auront une probabilité d'erreur triplée et l'on procède là encore à une méthode de Monte-Carlo pour différentes probabilités p_e.

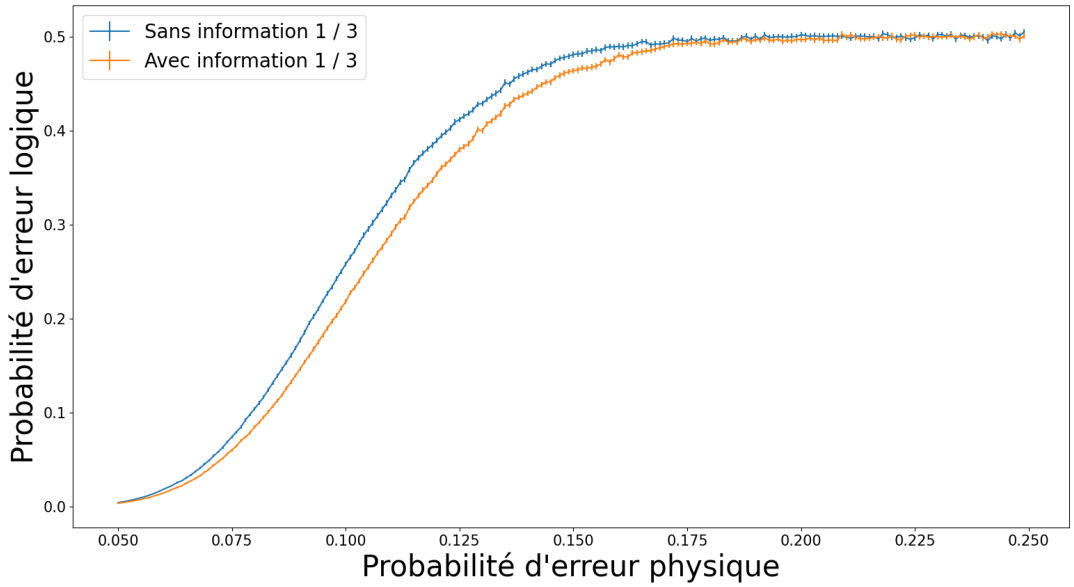


FIGURE 18 – Comparaison des performances du décodeur avec et sans prise en compte de l'information de la position des qubits davantage sujets aux erreurs.

Durant le décodage hiérarchique, on voudrait pouvoir adapter le décodage de plus haut niveau à partir des probabilités d'échec des décodages de plus bas niveau. Puisque que l'on connait le syndrome associé à chaque qubit logique intermédiaire, il semble possible de pouvoir utiliser cette information pour connaître approximativement cette probabilité d'échec. En simulant des erreurs et en regardant le taux de réussite pour chaque cardinal de syndrome, on pourrait en déduire une chance d'échec en corrélation avec ce même cardinal. La figure 19 montre cette probabilité d'échec en fonction de la taille du syndrome observé. De plus, la figure 20 montre bien la corrélation entre la taille du syndrome et la probabilité d'erreur sous-jacente. Il semble ainsi naturel de s'attendre à une probabilité d'erreur logique plus faible en cas de syndrome de plus petit cardinal.

En utilisant ces estimations pour améliorer le décodage hiérarchique de niveau supérieur, on obtient les résultats de la figure 21. Aucune différence notoire n'est discernable entre cette figure et la figure 17. La dernière modification semble donc inefficace, probablement car la corrélation entre la taille du syndrome et la probabilité d'erreur est assez moyenne et car la probabilité d'erreur estimée diffère peu d'un qubit logique intermédiaire à l'autre.

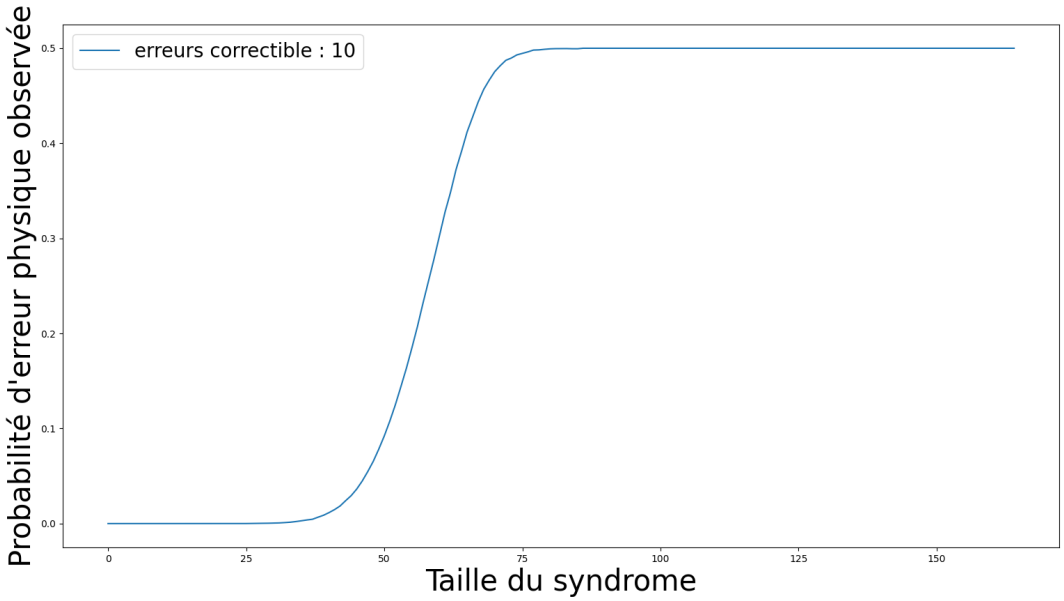


FIGURE 19 – Estimation de la probabilité d'erreur logique en fonction de la taille du syndrome pour un code couleur capable de corriger dix erreurs.

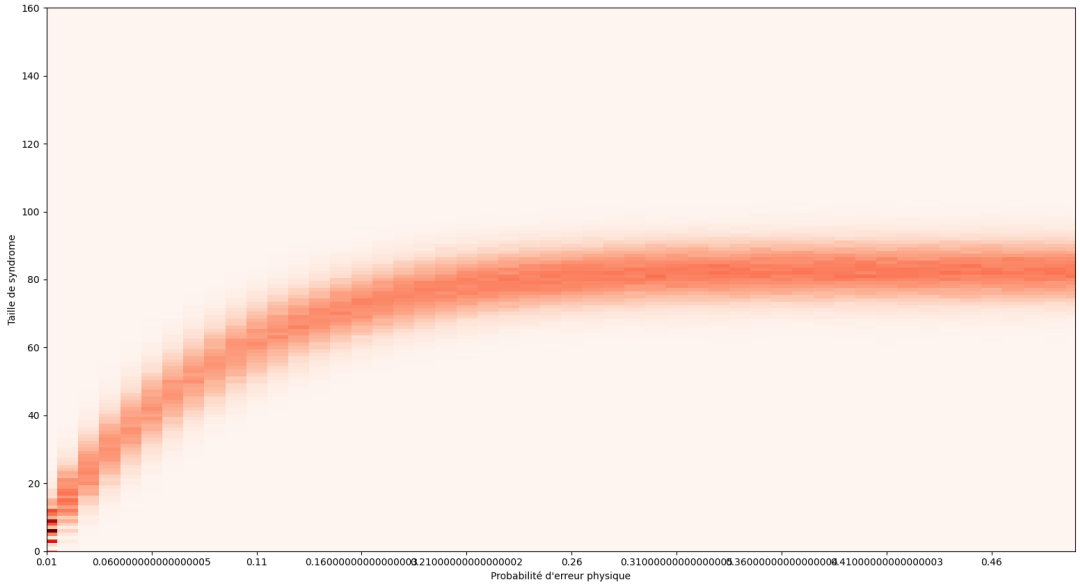


FIGURE 20 – Distribution empirique de la taille du syndrome en fonction de la probabilité d'erreur physique pour le code couleur capable de corriger dix erreurs.

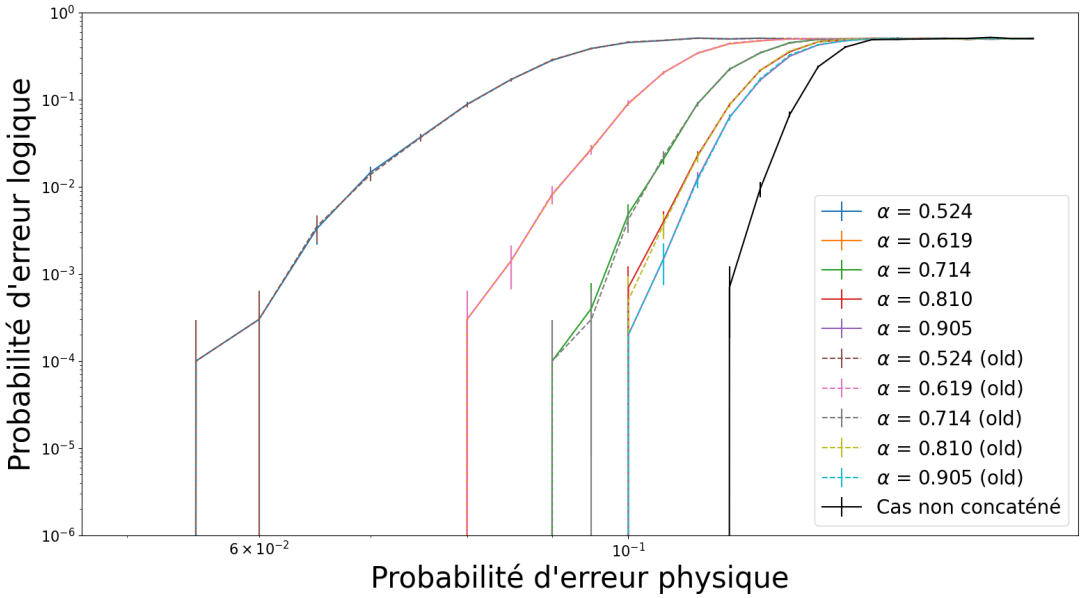


FIGURE 21 – Probabilté d'erreur logique à l'issue du décodage hiérarchique s'adaptant aux tailles des syndromes de plus bas niveaux estimée pour le décodage d'un code couleur capable de corriger dix erreurs concaténé avec lui même pour différents paramètres α.

4 Impacts environnementaux et sociétaux

4.1 Impact environnemental personnel du PFE

Mon impact environnemental personnel durant ce stage est assez faible. Sa tenue entièrement en télétravail a rendu les déplacements accomplis inexistants. De plus, le télétravail n'a pas induit de coûts supplémentaires, car les outils utilisés auraient été les mêmes en présentiel (ordinateurs, connexion internet, imprimante...) et ne m'ont de toute façon pas été fourni par le Loria.

Il est vrai que j'ai eu accès à l'infrastructure numérique du Loria (et de l'Inria), notamment pour l'adresse mail ainsi que pour les plateformes de discussions, au même titre que l'ensemble des collaborateurs de ces structures. Si on met de côté le coût fixe énergétique de ces infrastructures (non négligeable, difficile à estimer mais assez bien amorti par le nombre de projets), la création de mes comptes au sein de celles-ci n'a eu qu'un coût unitaire dérisoire et n'est donc pas à inclure dans mon impact personnel.

J'ai personnellement utilisé un ordinateur disposant d'une connexion internet durant huit heures en moyenne par jour, cinq jours par semaine durant vingt-six semaines. J'ai également utilisé une machine à distance pour les calculs de grande ampleur, elle est restée tout le temps allumée et est deux fois plus puissante que mon ordinateur. L'ordre de grandeur de la puissance consommée continuellement durant toute la durée du stage est donc de 10^2 W soit une consommation électrique totale de $10^2 \times 26 \times 7 \times 24 \approx 10^2$ kWh soit une émission de 10^1 kg eq CO_2 pour l'électricité française.

Le chauffage est généralement le processus le plus énergivore d'une infrastructure. Néanmoins, peu importe que le stage ait eu lieu en présentiel ou en télétravail, le chauffage du Loria et de ma chambre auraient fonctionné identiquement. Il n'est donc pas pertinent d'inclure le chauffage dans mon bilan personnel.

En conséquence, l'impact environnemental de mon stage a été d'environ 10^1 kg eq. CO_2, sans prendre en compte l'impact environnemental de fonctionnement du Loria et de ma maison, ni l'impact de la fabrication et du recyclage des outils utilisés, y compris le coût d'impression et d'envoi du présent rapport.

4.2 Impact global du projet

4.2.1 Impact environnemental global du projet

Le projet s'inscrit dans le cadre de la recherche publique mondiale, plus précisément dans le cadre de l'informatique quantique. Ce projet est de faible envergure au sein de celle-ci (petite équipe), consommant assez peu d'énergie (voir *supra*) et sans impact environnemental direct à moyen et long terme (une fois le projet clos et les articles scientifiques éventuellement publiés, l'impact direct devient nul).

La recherche en calcul quantique vise à pouvoir construire un ordinateur quantique suffisamment performant pour surpasser les supercalculateurs dans la résolution de certaines tâches complexes. La technologie étant encore embryonnaire, il n'y a aucune certitude sur le fait que le projet ait une quelconque application pratique un jour et certainement pas à court terme. Les effets rebonds sont donc particulièrement difficiles à concevoir car c'est l'avancée de la technologie des ordinateurs quantiques (en plus de l'intérêt pratique des résultats du projet) qui va dicter son utilisation réelle.

4.2.2 Impact sociétal global du projet

Là encore, l'absence de produit fini concrètement utilisé rend ardue (car prématurée) la discussion d'un impact sociétal discernable du projet. Aucune des dimensions listées dans les attendus de cette partie du rapport ne me semblent pertinente à mentionner pour l'impact du projet à son échelle.

4.2.3 Impact global de le recherche en calcul quantique

Les discussions de cette section sont pour l'instant largement insuffisantes par rapport aux critères d'évaluation. La nature du stage me semble inadaptée à une discussion à l'échelle du projet et nous allons nous placer au niveau plus large de la recherche en calcul quantique.

L'objectif du calcul quantique est de résoudre des problèmes difficiles même pour des supercalculateurs très énergivores (une dizaine de mégawatts dans le cas du Summit américain [13]) avec des dispositifs exploitant des propriétés quantiques de consommation plus modeste (quelques dizaines de kilowatts d'après certaines estimations [14]). Le gain énergétique virtuel est énorme, même si l'argument est fallacieux car les calculs classiques n'auraient jamais été entrepris d'une part et les consommations prévisionnels sont hautement incertaines d'autre part.

La chimie et la logistique sont parmi les domaines qui devraient bénéficier le plus de l'informatique quantique, ce qui expliquent que de nombreuses entreprises de ces secteurs investissent dans le calcul quantique[15]. L'optimisation des processus de production et de transport sont au cœur des applications envisagées, permettant idéalement une réduction drastique des coûts énergétiques et environnementaux d'une multitude de procédés industriels.

Le paradoxe de Jevons devrait s'appliquer de façon limitée au calcul quantique. En effet, en dépit d'une possible révolution dans la construction de machines quantiques toujours plus grandes, celles-ci devraient rester difficiles à manipuler, à entretenir et à rentabiliser. C'est pourquoi IBM s'attend à un marché mondial limité, et projette plutôt la mise à disposition, via le *cloud*, de temps de calcul quantique[16]. Si cette hypothèse de marché restreint s'avère exacte, toute amélioration technique réduisant les coûts (la correction d'erreurs quantiques vise à obtenir des résultats les plus fiables possibles en utilisant le moins de qubits possible et donc de coûts) est nécessairement écologiquement positive.

Du point de vue sociétal, la recherche – le calcul quantique ne faisant pas exception – peut avoir des conséquences tant positives que néfastes. D'une façon générale, la démarche scientifique correctement menée offre la possibilité à de nombreuses innovations d'émerger, contribuant à l'amélioration des conditions de vie de la population. Néanmoins, il n'est pas nécessaire de chercher beaucoup pour trouver des exemples de découvertes ayant conduit à des désastres sociétaux. La recherche scientifique ne saurait cependant être tenue responsable de l'utilisation maligne de ses fruits : Marie Curie n'a pas endeuillé le Japon, James Watt n'a pas causé le réchauffement climatique.

L'ordinateur quantique peut être vu comme une nouvelle génération d'ordinateurs ouvrant encore le champ des possibles. Tout comme les premiers ordinateurs ont été construits pour l'effort de guerre, il y a fort à parier que l'ordinateur quantique, s'il voit le jour, ait un rôle à jouer dans la géopolitique mondiale. Les perspectives qu'il offre pour le développement de technologies répondant aux problèmes d'aujourd'hui et demain sont cependant trop importantes pour être ignorées.

4.2.4 Nuances et critiques des points abordés

Jusqu'ici, j'ai supposé que l'ordinateur quantique serait une technologie fonctionnelle dans un futur proche et en accord avec la vision d'entreprises telles qu'IBM. Il est plausible que

celui-ci ne voit jamais le jour sous une forme telle qu'on le conçoit actuellement.

Supposons qu'il soit demandé la même analyse il y a cinquante ans à un chercheur travaillant sur le calcul tolérant aux fautes dans les ordinateurs classiques, dont les composants logiques étaient à l'époque sujets aux erreurs comme le sont les qubits aujourd'hui [17]. Les considérations écologiques étaient moindres à l'époque, mais l'escalade des armements atomiques faisait rage. Les mêmes promesses d'innovations formidables étaient proférées pour des futurs proches, de grands enjeux sociétaux étaient mis dans la balance.

Afin de considérer les conséquences de ses recherches, ce chercheur se serait certainement reposé sur la vision d'une entreprise alors en plein essor : IBM. IBM prévoyait déjà à l'époque un marché pour un nombre limité de machines, celles-ci étant alors énormes et extrêmement chères. Il aurait alors pu mettre de côté le paradoxe de Jevons comme conséquence de ses recherches pour les mêmes raisons que moi. Il aurait pu alors conclure sans crainte que la recherche en informatique serait sans grands impacts environnementaux dans les décennies qui allaient suivre.

Le développement de la physique quantique a permis la création de transistors beaucoup plus fiables, rendant d'une part le travail de ce chercheur inutile dans la création d'ordinateurs et d'autre part ses prévisions pour l'avenir de l'informatique et ses impacts aussi fausses que la vision d'IBM.

Cette analogie met bien en évidence qu'il est impossible de cerner l'impact global d'un pan de la recherche scientifique tant sur l'environnement que sur la société. L'analyse de la section précédente sera probablement rendue caduque et risible par les avancées qui seront faites dans les années à venir. Il est donc à mon sens préférable de s'abstenir d'écrire ce genre d'analyses tant que l'on reste dans le domaine de l'invention (la recherche ne pouvant prédire ce qu'elle va inventer) et de plutôt y recourir dans le cadre de l'innovation (l'entreprise pouvant dans une certaine mesure prévoir son impact et celui de ses produits sur l'environnement et la société). Cela ne signifie pas qu'il faut sacrifier toute considération écologique sur l'autel de l'avancée scientifique – il est bien entendu évident que celle-ci doit se poursuivre de la façon la plus durable possible – mais plutôt qu'il ne faut pas imputer à la recherche fondamentale les impacts de son utilisation par l'Homme à l'échelle mondiale.

4.3 Politique de la structure d'accueil

Le Loria est une institution publique à portée majoritairement locale, dont les locaux sont regroupés en un seul lieu. Les mesures prises s'appliquent donc à l'ensemble de l'entité et concernent tous les acteurs. Les mesures prises dans le cadre du développement durable au sein du Loria le sont sous l'égide de CARE (Commission pour l'Action et la Responsabilité Écologique). Le site internet consacré à ses actions détaille clairement les mesures prises avec leurs bénéfices bien établis triés suivant le pilier du développement durable qu'elles visent [18].

Les résultats chiffrés des actions sont également disponibles sur le site, bien que certaines actions mériteraient que l'on communique plus sur leur accomplissement et leurs résultats. Les objectifs que cette commission s'est fixés semble réalisable et ont l'avantage de ne reposer que sur des actions directes et concrètes au lieu de reposer sur la compensation carbone dont l'efficacité est aujourd'hui débattue.

Le Loria ne fournissant aucun produit ni service (autre que les contributions à la recherche et à l'éducation), il semble normal que les mesures prises ne concernent que le quotidien de celui-ci.

Un point négatif que je peux relever est l'absence de mention d'actions concernant le matériel informatique (ordinateurs de bureaux, serveurs) qui sont légion au Loria.

Conclusion

Ainsi, les différents objectifs initiaux n'ont été que partiellement remplis, non pas par oisiveté mais à cause des difficultés inhérentes aux domaines de recherche explorés. Néanmoins, j'estime avoir beaucoup et suffisamment appris sur les codes couleur et la correction d'erreurs pour pouvoir sereinement aborder la thèse qui va suivre ce stage sur un domaine connexe du calcul quantique, à savoir le calcul quantique tolérant aux fautes employant des codes de corrections d'erreurs dit de faible densité.

Au cours du stage, une bibliothèque Python permettant d'implémenter des codes couleur a vu le jour, ainsi qu'une façon de les concaténer topologiquement, ceci étant l'objet du stage. Les performances de cette bibliothèque sont assez bonnes étant donné qu'il est possible de simuler le décodage d'un code couleur comportant 10^7 qubits en quelques secondes. Néanmoins, le temps n'a pas permis d'implémenter les modèles d'erreurs raffinés, ni le processus de mesures des opérateurs de stabilisation qui serait requis pour ceux-ci.

Enfin, les conclusions quant aux performances du processus de concaténation topologique vis-à-vis de la concaténation classique ne sont guère définitives et demeurent astreintes à l'élaboration d'un décodeur plus performant pour les codes couleur en général.

Remerciements

Je voudrais tout d'abord remercier le Loria ainsi que les membres de l'équipe MOCQUA pour leur chaleureux accueil, légèrement terni par les conditions sanitaires ainsi que pour leurs remarques constructives. Nul doute que les échanges seront plus nombreux lorsque je serai sur place.

Je suis notamment reconnaissant à Simon Perdrix pour avoir répondu à ma sollicitation de stage au sein du Loria. S'agissant d'une candidature spontanée, ajoutant donc à sa charge de travail, il m'a aiguillé vers Christophe et vers mon stage au sein de l'équipe.

Emmanuel Jeandel fait également partie des membres de l'équipe qui mérite d'être amplement remercié ici : il a été présent pour répondre à mes questionnements et interrogations concernant la thèse et ses débouchés et c'est avec hâte que j'attends le début de celle-ci sous sa supervision.

Enfin, je voudrais tout particulièrement remercier Christophe Vuillot sans qui ce stage n'aurait pu être un succès. Il a su instaurer un climat propice à l'apprentissage et la découverte des codes quantiques correcteurs d'erreurs ainsi qu'à l'échange fructueux d'idées et d'objectifs. Toujours disponible, précis dans ses explications et l'exposé de ses attentes, il m'a régulièrement prodigué des conseils avisés. Il s'est également montré à l'écoute de mes remarques et de mes suggestions, permettant une avancée progressive et concertée du projet.

Glossaire

B

Bra : Dual d'un ket, satisfiant la relation : $\langle\phi| = |\phi\rangle^\dagger$.

C

Code de Hamming : Supposons que l'on souhaite envoyer quatre bits d'information $m = (1\,0\,1\,0)$, et que l'on dispose de sept bits pour les stocker. On peut utiliser un code de Hamming pour garantir la correction d'une erreur (et la détection d'une seconde) comme suit :

- On encode le message m grâce à la matrice génératrice :

$$x = G^T m = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

- Le message codé x est transmis et soumis aux erreurs. Supposons qu'on reçoive $y = (0\,1\,0\,1\,0\,0\,1)$. On vérifie la parité :

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

- On obtient un syndrome non nul, correspondant à une erreur sur le bit 5. On en déduit le message codé après correction $x' = (0\,1\,0\,1\,1\,0\,1)$ et de là le véritable message.

Codes de surface : Codes topologiques de correction d'erreurs plus simples que les codes couleur. Les opérateurs de stabilisation sont associés aux sommets et les qubits sont associés aux arêtes. Le processus de décodage se réalise comme suit :

- On récupère le syndrome i.e. la liste des sommets associés aux opérateurs de stabilisation insatisfaits par l'erreur.
- Au sein du graphe, on cherche un appariement deux à deux des sommets du syndrome. Deux sommets sont appariés via un chemin d'arêtes les reliant. L'appariement total utilisant le plus petit nombre d'arêtes est calculé (notamment par la bibliothèque `pymatching`).
- Les arêtes parcourues par cet appariement correspondent aux qubits qu'il faudra inverser pour corriger l'erreur (Fig. 22)

Coloriabilité : Un graphe planaire est dit n-coloriable s'il est possible d'associer à chaque face (ou dualement à chaque sommet) de celui-ci une des n couleurs de façon à ce que chaque face

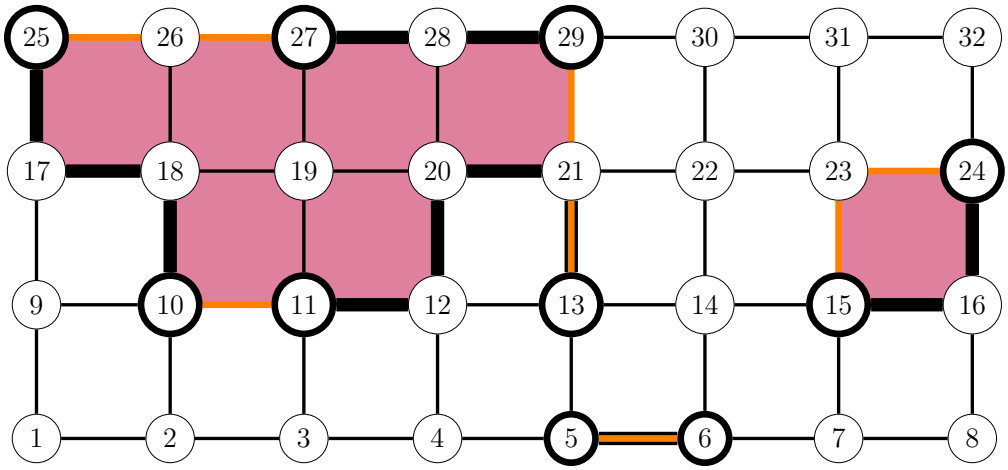


FIGURE 22 – Décodage et correction d'un code de surface. Les qubits erronés sont représentés par les épais traits noirs. Il engendre le syndrome représenté par les sommets dont le bord est également plus épais. L'appariement de poids minimal est indiqué en orange : il s'agit des qubits qui vont être inversés. On remarque que les arêtes de la différence symétrique entre l'erreur et la correction (les arêtes soit noires épaisses soit oranges) englobent des régions fermées (violet) garantissant l'absence d'erreurs logiques.

soit uniquement adjacente à des faces de couleurs différentes. Le fameux théorème des quatre couleurs affirme que tout graphe planaire est 4-coloriable.

D

Différence symétrique : Opération binaire sur les ensembles. La différence symétrique de deux ensembles A et B est un ensemble regroupant les éléments présents uniquement dans A ou dans B. On la note souvent $A \Delta B = A \cup B \setminus A \cap B$.

E

Effondrement (*collapse*) : Un qubit (ou tout état quantique) superposé dans la base dans laquelle il est mesuré voit son état s'effondrer vers l'un des kets propres de la base mesurée, et ce de façon aléatoire et pondérée par les coefficients de la combinaison linéaire. Par exemple, l'état $|\phi\rangle = |+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ est un état pur dans la base $(|+\rangle, |-\rangle)$. Sa mesure dans cette base donnera systématiquement 1 (valeur propre de $|+\rangle$) et l'état ne changera pas. À l'inverse, dans la base canonique $(|0\rangle, |1\rangle)$, la mesure de $|\phi\rangle$ donne 1 (valeur propre de $|0\rangle$) avec une probabilité $\frac{1}{2}$) et devient $|\phi'\rangle = |0\rangle$ ou la mesure de $|\phi\rangle$ donne -1 (valeur propre de $|1\rangle$) avec une probabilité $\frac{1}{2}$) et devient $|\phi'\rangle = |1\rangle$. On parle d'effondrement puisque la superposition d'états disparaît à l'issue de la mesure.
Espace de Hilbert : Espace vectoriel complet, muni d'un produit scalaire (hermitien dans le cas complexe) généralement noté $\langle\phi|\psi\rangle$.

G

Graphe dual : Le dual d'un graphe planaire est obtenu par construction comme suit :

- On attribue à chaque face (dont la face extérieure) un sommet
- Si deux faces du graphe initial sont adjacentes, on lie les deux sommets correspondants dans le graphe dual
- On obtient le graphe dual, dont chaque face est associable à un sommet dans le graphe primal

Pour construire le graphe dual du code couleur, on applique la procédure ci-dessus sur le graphe conservant le sommet qui a été ôté. En ôtant la face duale de ce sommet, on obtient trois sommets sur la frontière extérieure, qui correspondent aux trois frontières.

I

Intrication : Un état à plusieurs qubits est dit intriqué s'il n'est pas exprimable comme produit tensoriel d'états de qubit individuel. Un exemple simple est décrit à la figure 23. On applique d'abord H sur le premier qubit, donnant l'état $\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \otimes |0\rangle = \frac{|00\rangle + |10\rangle}{\sqrt{2}}$. On applique ensuite CNOT sur cet état, donnant ainsi l'état de Bell connu $\frac{|00\rangle + |11\rangle}{\sqrt{2}}$. La dépendance est ici évidente : la mesure du premier qubit va donner 0 avec une probabilité $\frac{1}{2}$ et 1 avec la même probabilité. Une fois cette mesure effectuée, il n'y a aucun doute possible sur la valeur du second qubit, et ce sans avoir besoin de le mesurer.

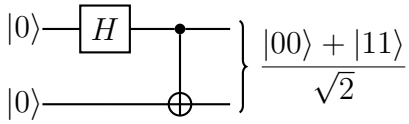


FIGURE 23 – Circuit quantique préparant un état de Bell intriqué

K

Ket : Vecteur d'un espace de Hilbert, généralement unitaire et noté $|\phi\rangle$.

O

Observable : Type d'opérateurs ayant la propriété d'être hermitiens, c'est-à-dire qu'ils sont égaux à leur transconjugué. Cette propriété leur garantit des valeurs propres réelles, donc accessibles à la mesure.
Opérateur : Application entre deux espaces vectoriels, possiblement identiques. Dans le cas d'espaces finis, un opérateur peut être mis sous une forme matricielle qui dépend des bases considérées pour les espaces de départ et d'arrivée.

P

Porte quantique : Opérateur unitaire physiquement implémenté dans une architecture d'ordinateur quantique, permettant l'évolution contrôlée de l'état du système de qubits dans le cadre de l'exécution d'un algorithme quantique. L'appelation "porte" provient de l'architecture classique des circuits, dont il existe un équivalent quantique. Par exemple, la figure 23 produit un état quantique intriqué à partir de deux portes quantiques. La figure 24 est un exemple de circuit plus élaboré, servant d'algorithme de transformée de Fourier quantique. Il comporte dix portes quantiques.
Produit tensoriel : Opération bilinéaire de deux espaces vectoriels vers un troisième, de dimension égale au produit des deux dimensions de départ. Moralement analogue à un produit cartésien dont on aurait réindexé les éléments tout en se conformant aux critères de linéarité. Pour les matrices, il se calcule comme suit :

$$A \otimes B = \begin{pmatrix} a_{11}B & \dots & a_{1m}B \\ \vdots & \ddots & \vdots \\ a_{n1}B & \dots & a_{nm}B \end{pmatrix}$$

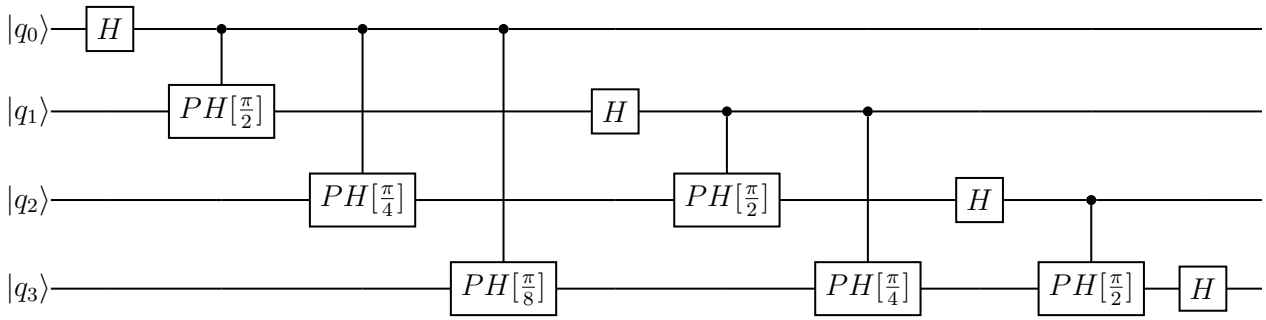


FIGURE 24 – Circuit quantique réalisant une transformée de Fourier quantique

S

Simplicité : Un graphe est simple si chaque paire de sommets distincts qu'il contient est reliée par au plus une arête.

T

Transconjuguée (†) : La matrice (ou le vecteur) $A^\dagger$ correspond à la transposée de la matrice contenant les complexes conjugués des éléments de A.

V

Valence : La valence d'un sommet est son nombre d'arêtes incidentes. Un graphe est dit n-valent si chacun de ses sommets possède une valence de n.

Références

[1] D. Gottesman, "Theory of fault-tolerant quantum computation," *Physical Review A*, vol. 57, p. 127–137, Jan 1998.

[2] C. Vuillot, *Fault-tolerant quantum computation : Theory and practice*. PhD thesis, Delft University of Technology, DOI:10.4233, 1 2020. Chapter 4.

[3] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information : 10th Anniversary Edition*. USA : Cambridge University Press, 10th ed., 2011.

[4] A. Kubica and M. E. Beverland, "Universal transversal gates with color codes : A simplified approach," *Physical Review A*, vol. 91, Mar 2015.

[5] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," *Physical Review A*, vol. 54, p. 1098–1105, Aug 1996.

[6] H. Bombin, "An introduction to topological quantum codes," 2013.

[7] A. Kubica and N. Delfosse, "Efficient color code decoders in $d \geq 2$ dimensions from toric code decoders," 2019.

[8] O. Higgott, "PyMatching : A python package for decoding quantum error correcting codes using minimum-weight perfect matching." `https://github.com/oscarhiggott/PyMatching`, 2020.

[9] K. M. Svore, D. P. DiVincenzo, and B. M. Terhal, "Noise threshold for a fault-tolerant two-dimensional lattice architecture," 2006.

[10] C. Chamberland, A. Kubica, T. J. Yoder, and G. Zhu, "Triangular color codes on trivalent graphs with flag qubits," *New Journal of Physics*, vol. 22, p. 023019, Feb 2020.

[11] E. Sabo, A. B. Aloshious, and K. R. Brown, "Trellis decoding for qudit stabilizer codes and its application to qubit topological codes," 2021.

[12] C. T. Chubb, "General tensor network decoding of 2d pauli codes," 2021.

[13] "Us dethrones china with ibm summit supercomputer." `https://www.tomshardware.com/news/us-supercomputer-china-top500-summit,37367.html`, 2018.

[14] "How much power will quantum computing need ?." `https://spectrum.ieee.org/how-much-power-will-quantum-computing-need`, 2015.

[15] `https://www.ibm.com/quantum-computing/case-studies/`, 2021.

[16] `https://www.ibm.com/quantum-computing/services/`, 2021.

[17] J. von Neumann, "Lectures on probabilistic logics and the synthesis of reliable organisms from unreliable components." lectures delivered at the California Institute of Technology, Notes by R. S. Pierce, 1952.

[18] `https://care.loria.fr/nos-missions/nos-actions/`, 2021.