

1 Première partie

Vous aurez à travailler sur trois instances du problème de Sudoku binaire et produire un rapport qui répond aux questions suivantes. Notez que nous ne travaillerons finalement pas la version 4x4 du Sudoku.

- Le rapport sera à remettre en pdf via Moodle d'ici le vendredi 6 décembre à 23h59.
- Le code doit être produit avec le package PennyLane et sera également à remettre sur Moodle.
- Il y aura un rapport par équipe de maximum 3 étudiant.e.s.
- Le rapport devrait inclure une page titre et répondre aux questions suivantes à l'aide de phrases complètes, de calculs et de graphiques.

1.1 Sudoku 2x2 - Version à 9 qubits

Codez le circuit à 9 qubits permettant de résoudre un sudoku binaire 2x2 à l'aide de l'algorithme de Grover.

- Expliquez comment on peut encoder la contrainte sur la première ligne à l'aide d'un circuit avec un qubit ancillaire.
- Expliquez le rôle de la porte CCCCNOT dans l'oracle.
- Pourquoi initialise-t-on le dernier qubit dans l'état $|-\rangle$?
- Quel est le nombre optimal d'itérations de V_f et W dans le circuit ? Présentez vos calculs.
- Quelle est la probabilité de succès ?
- Représentez graphiquement le circuit complet à simuler.
- Illustrez les probabilités de mesures en sortie du circuit.
- Combien de portes CNOT faut-il pour rouler le circuit sur une machine réelle ? (*Il est possible de faire ceci dans PennyLane, voir le fichier "Décompte de CNOT" sur Moodle.*)

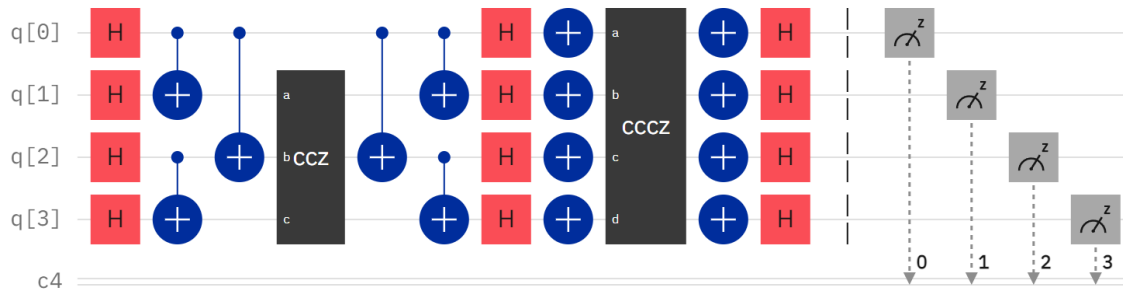
1.2 Sudoku 3x3 - Version à 16 qubits

Généraliser maintenant votre circuit pour le cas d'une grille de sudoku 3x3. Votre circuit devrait comporter 16 qubits.

- Combien y a-t-il de combinaisons possibles pour un sudoku 3x3 ?
- Combien y a-t-il de solutions possibles ?
- Quel est le nombre optimal d'itérations de V_f et W dans le circuit ? Présentez vos calculs.
- Quelle est la probabilité de succès ?
- Représentez graphiquement le circuit complet à simuler.
- Combien de portes CNOT faut-il pour rouler le circuit sur une machine réelle ?

1.3 Sudoku 2x2 - Version optimisée à 3 qubits

Nous avons vu en classe des optimisations permettant de réduire le circuit à 9 qubits pour un sudoku 2x2 au circuit suivant à seulement 4 qubits avec une seule itération de V_f et W . Nous utiliserons le quantum composer de IBM au lieu de PennyLane pour cette section pour faciliter le lancement de tâches sur des vraies machines quantiques.



- Créez le circuit à 4 qubits ci-haut sur le quantum composer de IBM et lancez la tâche sur une machine réelle. Présentez vos résultats et comparez-les à la probabilité de succès théorique.
- Vous devez maintenant réaliser le circuit optimisé à seulement 3 qubits où la première entrée v_0 fixée à 0. Représentez graphiquement le circuit complet à trois qubits à simuler pour une seule itération.
- En fixant $v_0 = 0$, combien y a-t-il de solutions possibles ?
- Quelle est la probabilité de succès théorique pour une itération de V_f et W ? Présentez vos calculs.
- Envoyez votre algorithme sur une vraie machine quantique à l'aide du quantum composer et présentez les résultats de mesures obtenus.
- Remplissez le tableau suivant en testant l'algorithme sur machine réelle avec les adaptations nécessaires :

Nombre d'itérations de $V_f W$	$P_{succ_théo}$	$P_{succ_réel}$	Nombre de CNOT dans le circuit
1			
2			
6			

N.B. Il est possible de décomposer le circuit en CNOT à l'aide d'un clique droit et de la commande "Ungroup" à l'intérieur du composer. Vous pourrez ainsi compter le nombre de CNOT nécessaires au circuit.

- Qu'en concluez-vous ? Comparez les résultats du tableau et commentez.
- Donnez le circuit à trois qubits à implémenter si on voulait maintenant obtenir les solutions avec $v_0 = 1$.
- (Bonus optionnel) Avez-vous des idées pour aller plus loin? Un bonus pourrait être accordé à la réalisation de tests supplémentaires vous permettant d'approfondir votre analyse.

2 Deuxième partie

N'oubliez pas d'inclure vos solutions des questions sur le swap-test et le codage superdense. Celles-ci peuvent être manuscrites et numérisées ou encore rédigées à l'aide d'un éditeur d'équations.