Projet Graphes et Optimisation Combinatoire

Qubit allocation problem

Jérôme Rouzé

Faculté polytechnique de Mons
Université de Mons

UMONS POLYTECH MONS

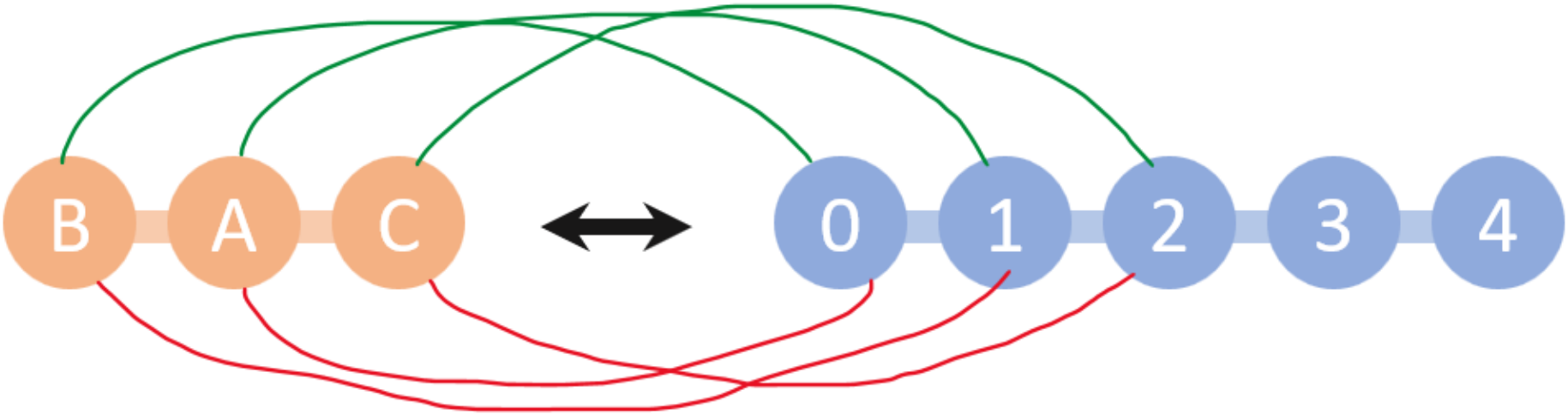
Présentation abstraite : problème de permutation partielle

Soit $n \leqslant m$ deux entiers naturels non nuls. On cherche le "meilleur" arrangement de n éléments parmi m.

- But du projet : Mettre un place un algorithme d'optimisation pour minimiser une fonction objectif.
- La fonction objectif : Prend en entrée une liste **ordonnée** de n entiers deux à deux distincts dans $\{0, \ldots, m-1\}$ et renvoie un coût.
- Exemple : avec $m = 5$, $n = 3$, $[0, 3, 1]$ et $[1, 0, 3]$ sont deux solutions différentes admissibles. $[0, 3, 3]$, $[0, 3]$, $[0, 3, 1, 4]$ ne le sont pas.

Les instances du problème et la fonction objectif seront fournies !

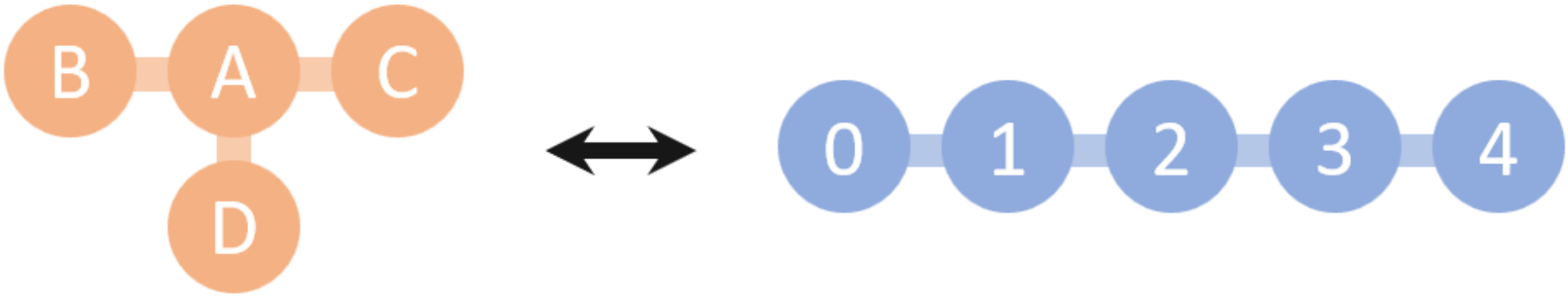
Application : Association de graphes



But : Associer les sommets du graphe orange à ceux du graphe bleu en préservant au mieux les arêtes oranges.

- Association $A \to 1; B \to 0; C \to 2$: aucun changement (association **parfaite**)
- Association $A \to 0; B \to 1; C \to 2$: plusieurs changements (association **imparfaite**)

Application : Association de graphes



But : Associer les sommets du graphe orange à ceux du graphe bleu en préservant au mieux les arêtes oranges.

- Pas d'association **parfaite**. Quelle est l'association qui minimise les changements (la moins imparfaite) ?

On attribue un coût à chaque changement et on cherche à minimiser le coût cumulé de chaque changement. Exemple d'application : le Qubit Allocation Problem.

Introduction : Qu'est-ce qu'un qubit ?

En informatique classique : un bit = 0 ou 1 ; deux bits : 00, 01, 10 ou 11.
En informatique quantique : un qubit = $\alpha|0\rangle + \beta|1\rangle$, où $\alpha, \beta \in \mathbb{C}$ tel que $|\alpha|^2 + |\beta|^2 = 1$
deux qubits = $\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$

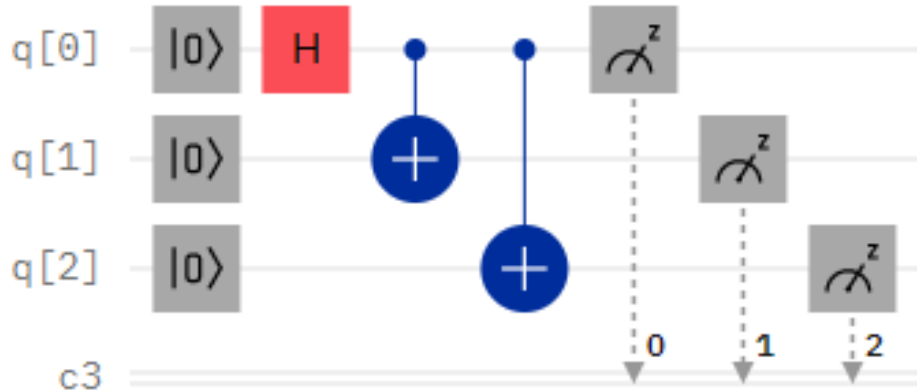
Deux propriétés quantiques :

- La superposition
- L'intrication (entanglement) : mathématiquement, c'est un état qui n'est pas le produit tensoriel de deux états. En pratique, c'est un état d'un système où plusieurs particules quantiques sont dans des états liés.

Exemple : Etat de Bell : $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$

Modèle à porte (gates-based model)

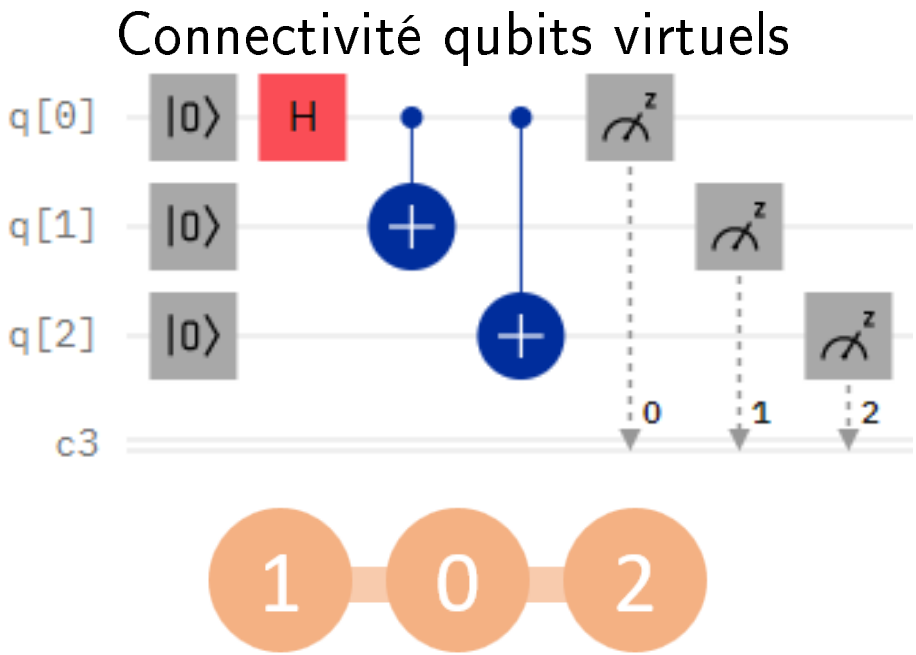
Dans ce modèle, un programme (ou circuit) quantique est une suite de portes quantiques ($\approx$ portes logiques). Un exemple de circuit :

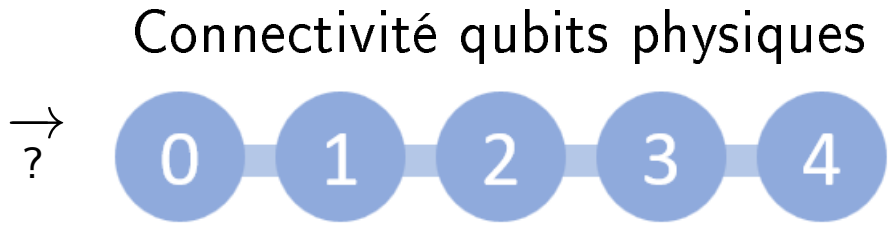


Porte H = porte de Hadamard, crée la superposition $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$

Porte en bleu = porte CNOT (Controlled NOT), applique NOT à la cible si le contrôle est en $|1\rangle$ et ne fait rien sinon. Crée de l'intrication.

Lien avec l'association de graphe

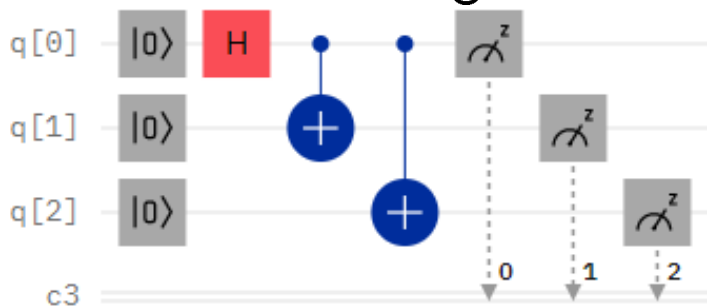




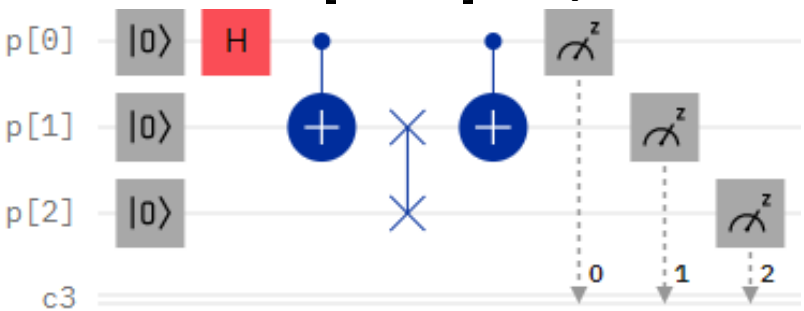
D'un circuit quantique, on tire un graphe orange à associer au graphe bleu, qui représente les connexions entre les qubits physiques d'un ordinateur quantique. Le coût des changements correspondent aux portes quantiques à ajouter au circuit pour l'exécuter malgré une association imparfaite.

Un exemple pour comprendre

Le circuit original

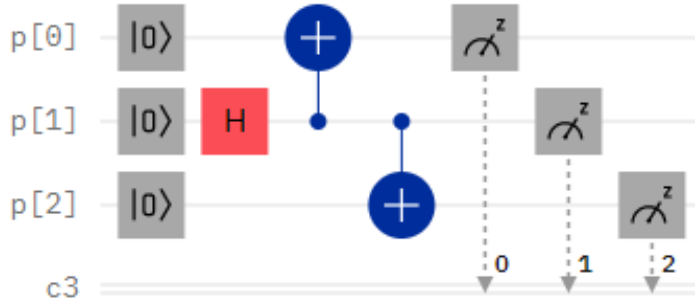


Association [0,1,2] **imparfaite**



Longueur = 6 (longeur(swap)=3)

Association [1,0,2] **parfaite**



Longueur = 3, solution optimale

Retour au projet

Sur Moodle se trouve le code StudentCode.py. Il fournie la fonction objectif et quelques fonctions utilitaires pour charger les instances du problème.

- La fonction objectif calcule les changements nécessaires et renvoie le coût. Elle prend en entrée **une liste ordonnée de n entier parmi** $\{0, \ldots, m-1\}$, n et m dépendant de l'instance (voir slide suivante).
- Le choix de l'instance se fait en changeant une variable.
- Ne rien changer d'autres !
- Pensez à installer Qiskit : `pip install qiskit`
- A la fin de l'execution de votre métaheuristique : écrire l'association puis son coût dans un ficher [GROUPE]_Instance_[instance_number].txt.

Exemple : Dans ROUZE_Instance_0.txt : 1 0 2 3

Utilisation de StudentCode.py

```
76  ##-------------------------------------------------------
77  ##     Pour choisir une instance:
78  ##     Modifier instance_num ET RIEN D'AUTRE
79  ##-------------------------------------------------------
80  instance_num=1     #### Entre 1 et 9 inclue
81
82  backend_name,circuit_type,num_qubit=instance_selection(instance_num)
83  backend,qc,qr=instance_characteristic(backend_name,circuit_type,num_qubit)
84
85  n=num_qubit
86  m=backend.num_qubits
87
88  ##-------------------------------------------------------
89  ##     A vous de jouer !
90  ##-------------------------------------------------------
91
92  ###### Votre code ici
```

En modifiant la variable instance_num, on change d'instance. Il ne faut rien modifier d'autres au dessus de l'espace "A vous de jouer". Les variables n et m qui dépendent de l'instance sont calculées, pas besoin de les entrer à la main !.

Les instances

Instance	n	m	Coût de la meilleur solution connue
1	20	27	47
2	20	127	46
3	27	27	67
4	27	127	68
5	20	127	57
6	27	27	90
7	20	127	23
8	27	127	30
9	14	27	43

Les instances se trouve dans le dossier "Instances" sur Moodle, à placer dans le même répertoire que StudentCode.py. StudentCode.py contient un test pour être sûr tout est correctement installé.