

Rapport
2023-2024

Sac à dos et machine quantique

Quentin Ambroziewicz
Ayoub Srij
Simon Demeilliers
Mohamed Aymen Hamid

JUNIA Grande
école
d'ingénieurs
HEI · ISEN · ISA



SOMMAIRE

Introduction	04
Formulation Mathématique	05
Notre Modèle	09
Les Solveurs	10
Utilisation de Gorubi	12
Application du Problème	14
Le Qubit	16
D-WAVE	17
MILP / QUBO	20
Projection Futures	22

INTRODUCTION

Le Problème du Sac à Dos (Knapsack Problem, KP) constitue un défi majeur en optimisation combinatoire, s'inscrivant à l'intersection de l'informatique et des mathématiques. Imaginons le dilemme du voyageur qui doit choisir judicieusement les objets à emporter dans son sac, chaque objet ayant un poids et une valeur spécifiques. Ce problème modélise de manière abstraite la nécessité de prendre des décisions optimales face à des ressources limitées.

Formulation du Problème : Le KP vise à déterminer la combinaison d'objets la plus précieuse à placer dans un sac à dos, compte tenu de contraintes de poids. Formellement, avec N objets caractérisés par leurs poids W_i et valeurs V_i , et un sac à dos de capacité maximale W , l'objectif est de maximiser la somme des valeurs des objets pris, tout en garantissant que le poids total ne dépasse pas la capacité du sac.

Complexité et Histoire : Le KP est réputé pour être un problème NP-complet, comme énoncé par Richard Karp en 1972 dans sa liste de 21 problèmes NP-complets. L'étude intensive de ce problème remonte au milieu du XXe siècle, avec des références dès 1897. Sa simplicité apparente contraste avec la complexité de sa résolution, faisant du KP un sujet privilégié de recherche.

Applications Réelles : Au-delà de son caractère académique, le KP a des applications concrètes étendues. Des algorithmes de résolution, tels que la programmation dynamique, les algorithmes gloutons et la programmation en nombres entiers, peuvent être appliqués à des problèmes du monde réel.

Ces applications vont de la préparation d'une valise pour un voyage à l'optimisation de la gestion des stocks dans des chaînes d'approvisionnement complexes.

Lien avec la Cryptographie : Le KP est également lié à l'histoire de la cryptographie. Il a servi de base au premier algorithme de chiffrement asymétrique, présenté par Hellman, Merkle et Diffie à Stanford en 1976. Bien que cet algorithme, connu sous le nom de Merkle-Hellman, ait été dépassé par RSA, il a marqué le début des recherches en cryptographie asymétrique.

FORMULATION MATHÉMATIQUE

Formulation Mathématique Détaillée du Problème du Sac à Dos :

Considérons le Problème du Sac à Dos (KP) avec N objets, chacun caractérisé par un poids W_i et une valeur V_i . Soit X_i une variable binaire indiquant si l'objet est inclus ($X_i=1$) ou non ($X_i=0$) dans le sac à dos. La capacité maximale du sac est W . On peut représenter par un vecteur X :

$$X = (x_1, x_2, \dots, x_n)$$

Fonction Objectif : L'objectif du KP est de maximiser la somme des valeurs des objets inclus dans le sac. Mathématiquement, cela se formule comme :

$$z(X) = \sum_{\{i, x_i=1\}} p_i = \sum_{i=1}^n x_i p_i$$

Cette fonction représente la valeur totale des objets sélectionnés.

Contrainte de Poids : La contrainte du poids s'exprime comme la somme des poids des objets sélectionnés ne doit pas dépasser la capacité du sac. Cela peut être formulé comme :

$$w(X) = \sum_{i=1}^n x_i w_i \leq W$$

Cette contrainte garantit que la somme des poids des objets dans le sac ne dépasse pas la capacité maximale autorisée.

L'objectif : Il faut réussir à utiliser les fonctions de la manière la plus efficace possible en terme de temps. Il est parfois préférable de négliger légèrement la qualité pour avoir un temps plus performant. On parle alors de la complexité qui va varier selon le programme utilisé. $O(nW)$ est la complexité en utilisant une fonction dynamique qui essaierait toutes les combinaisons possibles.

Avec n : nombre d'objet et W : poids du sac à dos

FORMULATION MATHÉMATIQUE

Formulation Complète : Ainsi, la formulation mathématique complète du KP est donnée par le problème d'optimisation suivant :

MAXIMISER $z(X)$:

$$z(X) = \sum_{\{i, x_i=1\}} p_i = \sum_{i=1}^n x_i p_i$$

SOUS LA CONTRAINTE DE POIDS $w(X)$:

$$: w(X) = \sum_{\{i, x_i=1\}} w_i = \sum_{i=1}^n x_i w_i$$

où x_i est une variable binaire indiquant si l'objet est inclus ($x_i=1$) ou non ($x_i=0$).

Contraintes suffisantes pour éviter les cas singuliers :

- $\sum_{i=1}^n w_i > W$: on ne peut pas mettre tous les objets ;
- $w_i \leq W, \forall i \in \{1, \dots, n\}$: aucun objet n'est plus lourd que ce que le sac peut porter ;
- $p_i > 0, \forall i \in \{1, \dots, n\}$: tout objet a une valeur et apporte un gain ;
- $w_i > 0, \forall i \in \{1, \dots, n\}$: tout objet a un certain poids et consomme des ressources.

Complexité et Applications : Le KP est classé comme un problème NP-Hard, ce qui implique qu'il peut être difficile à résoudre de manière exacte pour des instances de grande taille. Malgré cette complexité, le KP trouve des applications significatives dans des domaines tels que la gestion des stocks, la planification financière, la logistique, et l'optimisation des ressources.

Il est alors possible d'utiliser des solveurs adaptés pour les résoudre.

FORMULATION MATHEMATIQUE

Méthode exacte pour trouver la solution optimale :

Un arbre binaire peut être utilisé pour explorer toutes les combinaisons possibles d'objets que l'on peut mettre dans le sac à dos en respectant la contrainte de poids maximal. Voici un exemple d'arbre binaire pour résoudre ce problème :

Soit x_1, x_2, x_3 des variables binaires représentant respectivement si les objets 1, 2 et 3 sont pris ou non.

Les contraintes du problème sont :

- La capacité maximale du sac à dos est de 10 unités.
- Les poids des objets sont : $p_1=4$, $p_2=3$ et $p_3=4$.
- Les valeurs des objets sont : $v_1=7$, $v_2=4$ et $v_3=5$.

Nous cherchons à maximiser la valeur totale (V) tout en respectant la capacité du sac (P) :

$$V = v_1 \cdot x_1 + v_2 \cdot x_2 + v_3 \cdot x_3$$

$$P = p_1 \cdot x_1 + p_2 \cdot x_2 + p_3 \cdot x_3$$

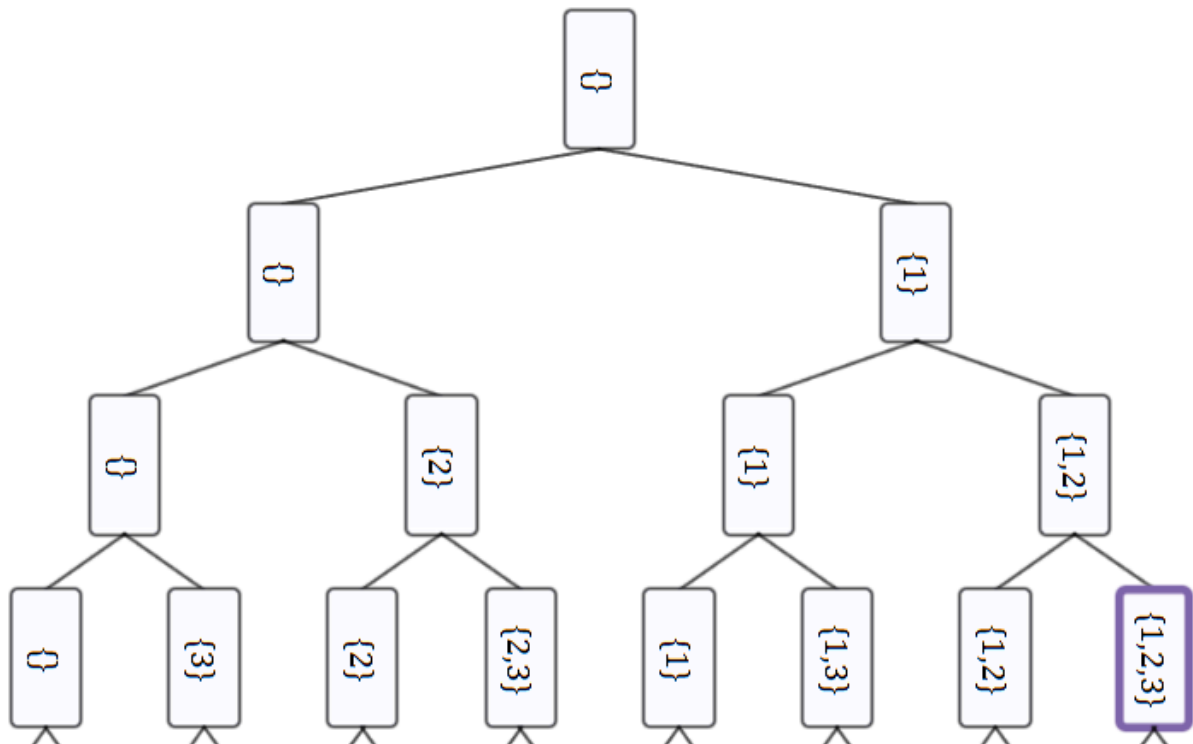
$$P \leq 10$$

Pour résoudre cela, on peut utiliser la méthode arborescente pour énumérer toutes les combinaisons possibles.

En effectuant les calculs :

1. $x_1=x_2=x_3=0$: $V=0$, $P=0$ (Racine)
2. $x_1=1, x_2=1, x_3=1$: $V=16, P=11$ (Limite de poids dépassée)
 3. $x_1=1 \ x_2=1 \ x_3=0$: $V=7$, $P=4$
 4. $x_1=1 \ x_2=0 \ x_3=1$: $V=12$, $P=8$
 5. $x_1=1 \ x_2=0 \ x_3=0$: $V=7$; $P=4$
 6. $x_1=0 \ x_2=1 \ x_3=1$: $V=9, P=7$
 7. $x_1=0 \ x_2=0 \ x_3=1$: $V=5$, $P=4$

FORMULATION MATHÉMATIQUE



En continuant ce processus, nous pouvons déterminer la meilleure combinaison qui maximise la valeur totale tout en respectant la capacité maximale du sac à dos. Dans cet exemple, la meilleure combinaison serait $x_1=1$ (Objet 1 pris) et $x_3=1$ (Objet 3 pris), avec une valeur totale de 12 et un poids total de 8, respectant la contrainte de poids maximal du sac à dos.

Une fois que l'arbre a été exploré (soit de manière récursive, soit à l'aide d'une autre méthode d'exploration), la solution optimale sera trouvée en choisissant le chemin qui maximise la valeur totale des objets tout en respectant la contrainte de poids.

Bien sûr, pour un grand nombre d'objets, cette méthode devient rapidement inefficace en raison de son coût exponentiel, mais elle illustre bien le principe de l'exploration binaire pour résoudre ce problème.

NOTRE MODÈLE

Application du Problème du Sac à Dos à la Sélection de Films pour une Clé USB :

Considérons le cas où nous avons une clé USB de 8 Go et une collection de 31 films, chacun avec un poids W_i (la taille du fichier en Go), une valeur V_i (Comprise entre 1 et 10) et X_i le choix de prendre ou non le film.

L'objectif est de maximiser la valeur totale des films que nous pouvons stocker sur la clé USB, en respectant la contrainte de poids imposée par la capacité maximale de la clé.

Données du Problème :

- Capacité de la clé USB (W) : 8 Go
- Nombre de films (N) : 31
- Le choix en binaire (X_i) : 0 ou 1
- Poids de chaque film (W_i) : donné en Go
- Valeur de chaque film (V_i) : peut être subjective, représentant par exemple la popularité ou le plaisir personnel

Numéro du film	Films (X_i)	Valeur (V_i) sur 10	Poids (W_i) en Go	Valeur / Poids (efficacité)
1	1	10	1	10
3	0	5	1,5	3,333333333
4	1	8	1,2	6,666666667
5	0	6	1,8	3,333333333
6	0	2	1,4	1,428571429
7	0	3	0,8	3,75
8	1	4	0,4	10
9	0	10	4	2,5
10	0	8	2	4
11	0	7	5	1,4
12	0	6	2,4	2,5
13	0	3	1	3
14	0	5	1,4	3,571428571
15	1	1	0,5	2
16	1	5	0,6	8,333333333
17	0	6	2,1	2,857142857
18	0	2	1,2	1,666666667
19	0	10	4	2,5
20	0	8	2,7	2,962962963
21	1	7	1,6	4,375
22	1	2	0,3	6,666666667
23	1	5	1	5
24	1	7	0,6	11,66666667
25	1	1	0,1	10
26	0	9	3,5	2,571428571
27	0	3	0,8	3,75
28	0	1	0,5	2
29	0	4	1,5	2,666666667
30	0	6	2	3
31	0	8	2,4	3,333333333
			poids total:	50

LES SOLVEURS

Choix de Solveurs pour un Projet d'Optimisation :

Dans le domaine de l'optimisation, le choix du solveur est crucial pour la résolution efficace des problèmes. Deux des solveurs commerciaux les plus renommés, CPLEX et Gurobi, sont souvent considérés comme les meilleurs dans le domaine. Cependant, la disponibilité gratuite de versions académiques peut influencer la décision, en particulier pour des projets de recherche ou éducatifs.

- **CPLEX :**
 - Disponibilité Académique : Gratuit pour les étudiants et la communauté académique.
 - Réputation : CPLEX, développé par IBM, est reconnu pour sa puissance dans la résolution de problèmes d'optimisation linéaire, quadratique et mixte-entier.
 - Communauté : Bénéficie d'une grande communauté d'utilisateurs et de support.
 - Interface : Dispose d'une interface utilisateur conviviale et de nombreuses options de programmation.
- **Gurobi :**
 - Disponibilité Académique : Gratuit pour les étudiants et les enseignants dans un cadre académique.
 - Performance : Gurobi est réputé pour sa rapidité d'exécution et a gagné en popularité ces dernières années.
 - Fonctionnalités : Propose des fonctionnalités avancées telles que la parallélisation.
 - Interface : Dispose d'une interface utilisateur intuitive et d'une documentation complète. Facilité d'utilisation grâce à la librairie python

LES SOLVEURS

Solveurs Libres :

- **GLPK (GNU Linear Programming Kit) :**
 - Gratuité : Open-source et donc gratuit.
 - Fonctionnalités : Convient pour des problèmes de programmation linéaire et mixte-entier.
 - Communauté : Communauté active, mais peut ne pas être aussi étendue que celle des solveurs commerciaux.
- **COIN-OR :**
 - Flexibilité : Ensemble de bibliothèques open-source pour l'optimisation, avec divers solveurs inclus.
 - Puissance : Plus puissant que certains autres solveurs libres, mais peut nécessiter une configuration spécifique.
 - Formats de Fichier : Gère généralement les fichiers .lp, mais pas évident d'utilisation

Choix Stratégique :

Le choix de Gurobi comme solveur a été effectué car voulant utiliser du Python , étant un langage facile d'utilisation et connu de chaque membre du groupe. La librairie gurobipy est très complète et permet l'utilisation de fichier .lp. Le fait de pouvoir obtenir une licence académique est également très intéressant et après quelques hésitations entre CPLEX et Gurobi On a choisi Gurobi car depuis quelques années il se montre de plus en plus performant et se place même premier en terme de vitesse d'exécution donc nous souhaitons utiliser le logiciel le plus performant. La documentation est très complète sur : <https://www.gurobi.com/documentation>

UTILISATION DE GORUBI

Demande de licence académique :

Après avoir effectué la demande de la licence académique du Gorubi, il a été possible d'installer Gorubi Optimizer.

Installation :

En terme d'installation, l'utilisation de Anaconda est pertinente car elle fournit la distribution de Python et donne accès à un environnement virtuel permettant d'écrire le script et de charger des données, notamment la lecture d'un fichier.lp

Les fichiers .lp :

Le format .lp est utilisé pour faire la programmation linéaire : étant un fichier permettant de construire le modèle mathématique de façon très lisible et simplifié. Cependant , il y a quelques défauts à l'utilisation de ce modèle comme le fait que l'ordre des colonnes n'est généralement pas préservé. Cela signifie que la disposition des variables dans le fichier LP peut ne pas être conservée lors de la lecture du fichier, ce qui peut rendre difficile la compréhension du modèle. Il y à également le fait que le .lp ne peut pas être utilisé pour tous les problèmes car certaines formes complexes ne peuvent être exprimés de manière simplifié.

Exemple de fichier .lp pour notre modèle :

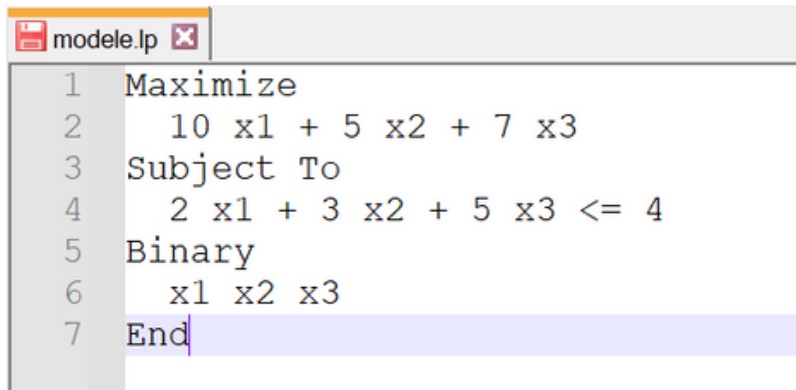
Supposons que nous ayons 3 films avec les caractéristiques suivantes :

- Clé USB de 4 Go
- Film 1 : Poids = 2 Go, Valeur = 10
- Film 2 : Poids = 3 Go, Valeur = 5
- Film 3 : Poids = 5 Go, Valeur = 7

UTILISATION DE GORUBI

Exemple du fichier .lp associé :

Il s'agit d'un exemple trivial mais permettant de comprendre le fonctionnement du fichier .lp car l'utiliser pour 3 films ou 10000 films reste le même fonctionnement.



```
1 Maximize
2   10 x1 + 5 x2 + 7 x3
3 Subject To
4   2 x1 + 3 x2 + 5 x3 <= 4
5 Binary
6   x1 x2 x3
7 End
```

Explications :

- La ligne "Maximize" permet de donner l'information sur les valeurs à Maximiser : Fonction Objectif
- La ligne "Subject To" définit les poids avec la contrainte de Poids :
- Sous la contrainte : Fonction Contrainte
- "Binary" indique que les variables de décision (x1, x2, x3) sont binaires, car chaque article peut être pris ou non (0/1).
- "End" marque la fin du fichier .lp.

Réalisation :

Création du fichier .lp à l'aide d'un programme python puis résolution avec Gorubi

APPLICATION DU PROBLÈME

Nous allons à présent résoudre notre problème à l'aide de gurobi et Excel.

Tout d'abord, Excel nous permet de choisir un solveur linéaire capable de résoudre des problèmes d'optimisation tant que le nombre de variables n'est pas trop élevé. Les temps d'exécution peuvent cependant rapidement devenir long.

Numéro du film	Films (Xi)	Valeur (Vi) sur 10	Poids (Wi) en Go	Valeur / Poids (efficacité)	Valeur ligne (Xi*Vi)	Valeur sac à dos(Somme des Xi*Vi)	poids ligne(Xi*Wi)	poids sac à dos(somme des Xi*Wi)
1	1	10	1	10	66	1	7,8	
2	1	7	1,1	6,3636364	7	1,1		
3	1	5	0,5	10	5	0,5		
4	1	8	1,2	6,6666667	8	1,2		
5	1	6	0,8	7,5	6	0,8		
6	0	2	1,4	1,428571429	0	0		
7	0	3	1,8	1,6666667	0	0		
8	0	4	1	4	0	0		
9	1	10	0,3	33,3333333	10	0,3		
10	0	8	2	4	0	0		
11	0	7	5	1,4	0	0		
12	0	6	2,4	2,5	0	0		
13	0	3	1	3	0	0		
14	0	5	1,4	3,571428571	0	0		
15	0	1	1,5	0,6666667	0	0		
16	0	5	1,6	3,125	0	0		
17	0	6	2,1	2,857142857	0	0		
18	1	2	0,2	10	2	0,2		
19	0	10	4	2,5	0	0		
20	0	8	2,7	2,962962963	0	0		
21	1	7	1,6	4,375	7	1,6		
22	0	2	1	2	0	0		
23	0	5	1	5	0	0		
24	1	7	0,6	11,6666667	7	0,6		
25	0	1	1,1	0,909090909	0	0		
26	0	9	3,5	2,571428571	0	0		
27	0	3	1,8	1,6666667	0	0		
28	0	1	1,5	0,6666667	0	0		
29	1	4	0,5	8	4	0,5		
30	0	6	2	3	0	0		
31	0	8	2,4	3,3333333	0	0		

Ici, on remarque que le solveur Excel obtient une valeur maximale de 66 points quand le disque possède une capacité de 8Go. Ensuite, nous avons créé un script python qui transforme un fichier .csv (Excel) en un fichier .Lp lisible par le solveur gurobi et nous avons résolu le même problème afin de vérifier notre valeur maximale.

APPLICATION DU PROBLÈME

```
Optimize a model with 1 rows, 31 columns and 31 nonzeros
Model fingerprint: 0x6ce0f9c0
Variable types: 0 continuous, 31 integer (31 binary)
Coefficient statistics:
  Matrix range      [2e-01, 5e+00]
  Objective range   [1e+00, 1e+01]
  Bounds range      [1e+00, 1e+00]
  RHS range         [8e+00, 8e+00]
Found heuristic solution: objective 43.0000000
Presolve removed 0 rows and 2 columns
Presolve time: 0.00s
Presolved: 1 rows, 29 columns, 29 nonzeros
Variable types: 0 continuous, 29 integer (27 binary)
Found heuristic solution: objective 66.0000000

Root relaxation: objective 6.750000e+01, 1 iterations, 0.00 seconds (0.00 work units)
...
x27 = 0.00
x28 = 1.00
x29 = 0.00
x30 = 0.00
```

Gurobi nous donne lui aussi une valeur de 66 ce qui permet de confirmer que les deux solveurs sont bien configurés. Cependant, le programme que nous avons fait nécessite de créer des fichiers .csv à la main ce qui serait très long pour résoudre des problèmes avec un nombre important de variables. Nous avons donc créé un nouveau programme qui permet de créer un fichier avec beaucoup de variables (ici 5000) afin de tester la rapidité du solveur. Tout d'abord on remarque que le solveur utilise une partie plus importante de la capacité de calcul de notre ordinateur que pour notre exemple, mais la vitesse de résolution reste presque nulle ce qui montre que le solveur est très performant.

```
Optimize a model with 1 rows, 5000 columns and 5000 nonzeros
Model fingerprint: 0x94b341ae
Variable types: 0 continuous, 5000 integer (5000 binary)
Coefficient statistics:
  Matrix range      [9e-04, 5e+00]
  Objective range   [1e+00, 1e+01]
  Bounds range      [1e+00, 1e+00]
  RHS range         [8e+00, 8e+00]
Found heuristic solution: objective 67.0000000
Presolve time: 0.04s
Presolved: 1 rows, 5000 columns, 5000 nonzeros
Variable types: 0 continuous, 5000 integer (5000 binary)
Found heuristic solution: objective 124.0000000

Root relaxation: objective 7.640099e+02, 1 iterations, 0.00 seconds (0.00 work units)
```

LE QUBIT

Dans un ordinateur traditionnel, les données sont stockées sous forme de bits dans des registres, chaque bit ayant une valeur de 0 ou 1. En revanche, un qubit (bit quantique) possède deux états quantiques, notés $|0\rangle$ et $|1\rangle$, séparés par une différence d'énergie qui définit sa fréquence (fQB).

De manière unique, un qubit peut simultanément se trouver dans ces deux états. Lors de l'exécution d'un algorithme, qui consiste en une séquence d'opérations appelées « portes logiques », le registre de qubits se place dans une superposition quantique englobant tous les états possibles ($|00\dots0\rangle$, $|10\dots0\rangle$, $|11\dots1\rangle$, $|10\dots1\rangle$). Cette superposition permet d'effectuer des calculs de manière massivement parallèle.

En raison de ses caractéristiques quantiques telles que la superposition et l'intrication, un registre composé de N qubits se trouve simultanément dans une superposition de ses 2^N configurations de base à un instant donné. En revanche, un registre constitué de N bits classiques ne peut occuper qu'une seule de ces configurations à la fois.

Cela signifie que toute opération appliquée à un registre de N qubits se déroule en parallèle sur les 2^N états, offrant ainsi un avantage significatif par rapport à un ordinateur classique qui doit traiter les opérations de manière séquentielle. Ce potentiel de parallélisme massif ouvre des perspectives extrêmement prometteuses, laissant entrevoir la possibilité de résoudre certains problèmes beaucoup plus rapidement ou de trouver des solutions à des défis actuellement considérés comme insolubles.

Cependant, de nombreux défis d'ordre physique et technologique entravent le développement de l'informatique quantique, en commençant par la fragilité inhérente à l'état de superposition requis.



D-WAVE

D-Wave Systems est une entreprise canadienne fondée en 1999 par Geordie Rose, Haig Farris et Bob Wiens. Initialement, l'objectif était de développer des ordinateurs quantiques pour résoudre des problèmes complexes et ouvrir de nouvelles perspectives dans le domaine de l'informatique.

D-Wave se positionne comme l'un des pionniers dans le domaine de l'informatique quantique. L'entreprise se concentre sur la fabrication et la commercialisation d'ordinateurs quantiques. Sa technologie repose sur des processeurs quantiques, appelés des processeurs d'unité de traitement quantique (QPU), conçus pour effectuer des calculs complexes en exploitant les principes de la mécanique quantique.

Les ordinateurs quantiques diffèrent fondamentalement des ordinateurs classiques en utilisant des qubits, les unités de stockage de données quantiques. Contrairement aux bits classiques qui ne peuvent être que 0 ou 1, les qubits peuvent représenter des superpositions de 0 et de 1 simultanément, offrant ainsi un énorme potentiel pour traiter des problèmes complexes de manière exponentielle plus rapide que les ordinateurs conventionnels.

Les machines de D-Wave sont utilisées dans des domaines variés tels que l'optimisation, la recherche en intelligence artificielle, la chimie quantique et même dans des applications de sécurité et de cryptographie. Les ordinateurs quantiques promettent de révolutionner de nombreux secteurs en résolvant des problèmes insolubles pour les ordinateurs classiques. Ils pourraient accélérer la découverte de nouveaux médicaments, optimiser les chaînes logistiques, améliorer les modèles météorologiques et offrir des avancées significatives dans le domaine de l'intelligence artificielle.

D-WAVE

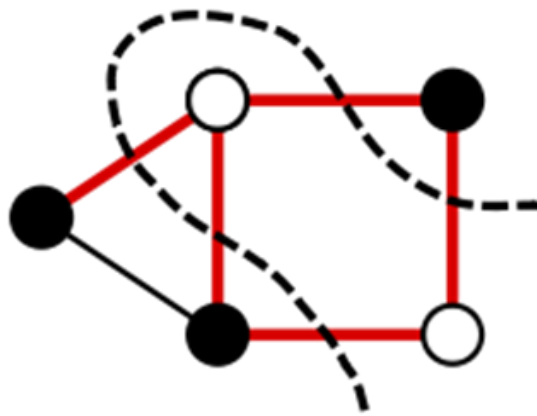
Bien que prometteurs, les ordinateurs quantiques ne sont pas encore capable de rivaliser avec les ordinateurs classiques. Avec 5000 qubits, il s'agit seulement de résolution de problème d'optimisation sans l'utilisation de porte logique comme sur un ordinateur classique ou sur la machine quantique de IBM.

IBM possédant sa machine avec 450 qubits rivalise largement avec D-Wave à travers le fait qu'il s'agisse d'une machine universelle donc pouvant fonctionner comme un ordinateur classique. L'avenir se portera donc vers IBM même si pour faire de l'optimisation, D-Wave s'impose comme leader actuel. L'avantage de D-Wave est donc une machine de choix pour l'étude des problèmes d'optimisations comme le knapsack ou le Max cut.

Le problème de maximum cut est un problème d'optimisation dans la théorie des graphes. Prenons une figure avec n sommets, la coupe maximum est la coupe qui contient le plus d'arêtes différentes.

On peut aussi donner un poids à chaque arête et la coupe maximum est la coupe qui possède la somme des poids la plus importante. Ce problème peut être résumé sous forme d'équation :

On donne aux arêtes (x_i, x_j) du graphe les valeurs 1 si elles ne sont pas coupées ou -1 si elles sont coupées. On souhaite ensuite minimiser l'équation. Le but est donc toujours d'avoir l'un des 2 à -1 afin d'obtenir une séparation optimale. Le problème est NP-hard ce qui signifie que la complexité est exponentielle lorsque le nombre d'arêtes augmente.



D-WAVE

Accès à la machine quantique de D-WAVE par le Cloud :

Il est possible d'utiliser le service de cloud de D-WAVE via leur site web : <https://cloud.dwavesys.com/leap/login/?next=/leap/> avec un identifiant pour pouvoir l'utiliser.

Les utilisateurs doivent créer un compte sur la plateforme D-WAVE Leap, qui est le service cloud de D-WAVE. Une fois inscrits, les utilisateurs peuvent exécuter des programmes quantiques.

Les utilisateurs peuvent écrire des programmes quantiques en utilisant le langage de programmation Ocean, qui est une suite d'outils et de bibliothèques fournis par D-Wave pour travailler avec leurs systèmes quantiques. La programmation se fait généralement en Python, et les programmes peuvent définir des problèmes d'optimisation sous forme de QUBO (Quadratic Unconstrained Binary Optimization) ou d'Ising.

Accès à la machine quantique de D-WAVE en local :

Il suffit de s'inscrire sur le site de D-WAVE pour obtenir les informations et les outils nécessaires. Il faut installer les outils Ocean de D-WAVE, qui comprennent les bibliothèques Python et les utilitaires nécessaires pour programmer et exécuter des tâches quantiques.

L'installation se fait avec : `pip install dwave-ocean-sdk`

Puissance de D-WAVE :

	2000Q	Advantage	Advantage performance update
Performance			
Better Solutions (Satisfiability problems)	--	3x more often than 2000Q	23x more often than 2000Q
Time-To-Solution (3D lattice problems)	--	10x faster than 2000Q	2x faster than Advantage
Annealing Quantum Processor Design			
Qubits	2000+	5000+	5000+
Couplers	6000+	35000+	35000+
Couplers Per Qubit	6	15	15
Topology			
Graph	Chimera	Pegasus	Pegasus
Graph Size	C16	P16	P16
Connectivity	Degree 6	Degree 15	Degree 15
Lattice	8x8x8	15x15x12	15x15x12
Chain Length (for problem size n=64)	17	7	6

MILP / QUBO

Le Mixed Integer Linear Programming (MILP) est une méthode d'optimisation utilisée pour résoudre des problèmes qui impliquent à la fois des variables continues et discrètes. Dans un problème MILP, certaines variables peuvent prendre des valeurs entières tandis que d'autres sont continues. Il vise à maximiser ou minimiser une fonction objectif tout en respectant un ensemble de contraintes linéaires.

L'objectif du MILP est de trouver les valeurs des variables qui optimisent une fonction objectif sous des contraintes données. Ce type de problème peut être formulé mathématiquement comme suit :

$$\text{Minimiser } \mathbf{c}^T \mathbf{x} + \mathbf{d}^T \mathbf{y}$$

sous contraintes:

$$\mathbf{Ax} + \mathbf{By} \leq \mathbf{b}$$

$$\mathbf{x} \in \mathbb{R}^n, \mathbf{y} \in \mathbb{Z}^m$$

où \mathbf{x} et \mathbf{y} sont les vecteurs des variables continues et entières respectivement, \mathbf{c} et \mathbf{d} sont les vecteurs des coefficients de la fonction objectif, \mathbf{A} et \mathbf{B} sont les matrices des coefficients des contraintes, et \mathbf{b} est le vecteur des constantes de contrainte.

Le QUBO, ou Quadratic Unconstrained Binary Optimization, est un formalisme mathématique utilisé pour résoudre des problèmes d'optimisation. Il se concentre sur la minimisation d'une fonction objectif qui est exprimée de manière quadratique et basée sur des variables binaires. Ces variables prennent des valeurs de 0 ou 1, ce qui représente souvent des états ou des décisions dans diverses applications.

MILP / QUBO

Le problème QUBO (Quadratic Unconstrained Binary Optimization) est un problème NP-difficiles. Cela signifie qu'il fait partie des problèmes pour lesquels la vérification d'une solution est réalisable en temps polynomial, mais la découverte d'une solution optimale nécessite potentiellement un temps exponentiel en fonction de la taille du problème.

D'un point de vue définition, on travaille avec un ensemble de vecteurs binaires de longueur fixe $n > 0$, représenté par B_n . Cet ensemble est composé de valeurs binaires, généralement notées $\{0,1\}$, représentant les bits. Pour formuler ce problème, une matrice Q de taille $n \times n$, de valeurs réelles, est utilisée. Les entrées de cette matrice, Q_{ij} , définissent les poids pour chaque paire d'indices i et j dans le vecteur binaire.

Il est possible de le formuler avec le modèle d'Ising, on parle alors que de valeurs notées $\{-1 ; 1\}$

À l'aide de cette matrice, une fonction f_Q est définie pour attribuer une valeur à chaque vecteur binaire. Cette fonction s'exprime comme

$$f_Q(x) = x^T Q x = \sum_{i=1}^n \sum_{j=i}^n Q_{ij} x_i x_j$$

Le problème QUBO vise à trouver un vecteur binaire qui minimise la fonction f_Q parmi tous les vecteurs binaires possibles.

La complexité du QUBO réside dans le nombre exponentiel de vecteurs binaires à évaluer, car

$$|B^n| = 2^n \text{ croît exponentiellement avec } n.$$

Parfois, le QUBO est formulé comme un problème de maximisation $f-Q=-f_Q$, ce qui revient à maximiser la fonction f_Q plutôt que de la minimiser.

PROJECTIONS FUTURES

Réalisation:

Après avoir achevé la phase de recherche initiale et la mise en place des bases théoriques nécessaires, notre projet se tourne maintenant vers des applications pratiques. Cette section détaille la feuille de route pour les prochaines étapes du projet, incluant l'installation du logiciel D-Wave et l'application de ce dernier pour résoudre le problème du sac à dos.

Objectifs:

1. Installer et configurer le logiciel D-Wave.
2. Formuler le problème du sac à dos en format QUBO.
3. Programmer et tester la solution sur le système D-Wave.
4. Analyser les résultats et les comparer avec des solutions classiques.

Étapes détaillées:

Installation du Logiciel D-Wave:

- Sous-étape 1: Configuration de l'environnement de développement.
- Sous-étape 2: Installation des bibliothèques et dépendances.

Formulation et Programmation:

- Sous-étape 1: Transformation du problème du sac à dos en QUBO.
- Sous-étape 2: Écriture du code pour l'implémentation sur D-Wave.

Tests et Optimisation:

- Sous-étape 1: Exécution des premiers tests sur le système D-Wave.
- Sous-étape 2: Ajustement des paramètres pour optimiser les résultats.

Analyse des Résultats:

- Sous-étape 1: Collecte et évaluation des données de sortie.
- Sous-étape 2: Comparaison avec les méthodes d'optimisation classiques.

SOURCES

https://www.gurobi.com/documentation/current/refman/lp_format.html

https://fr.wikipedia.org/wiki/Probl%C3%A8me_du_sac_%C3%A0_dos

https://www.ibm.com/docs/fr/icos/12.10.0?topic=SSSA5P_12.10.0

<https://www.gnu.org/software/glpk/>

<https://www.coin-or.org/resources/>

<https://dept-info.labri.fr/ENSEIGNEMENT/projet2/supports/Sac-a-dos/Le-probleme-du-sac-a-dos.pdf>

<https://www.youtube.com/watch?v=fQVxuWOiPpl> (installation + anaconda)

<https://fr.wikipedia.org/wiki/D-Wave>

https://fr.wikipedia.org/wiki/Coupe_maximum

<https://www.dwavesys.com/solutions-and-products/systems/>

<https://quantumcomputing.stackexchange.com/questions/26394/how-are-ibms-127-qubits-more-potent-than-the-5760-qubits-d-wave-advantage-sys>

[https://www.ccea.fr/comprendre/Pages/nouvelles-technologies/essentiel-sur-ordinateur-quantique.aspx](https://www cea.fr/comprendre/Pages/nouvelles-technologies/essentiel-sur-ordinateur-quantique.aspx)

REMERCIEMENT

Nous tenons à remercier Samuel DELEPLANQUE pour la proposition du sujet mais également l'accompagnement et les explications , informations pertinentes qui nous ont permis une meilleure compréhension du sujet.