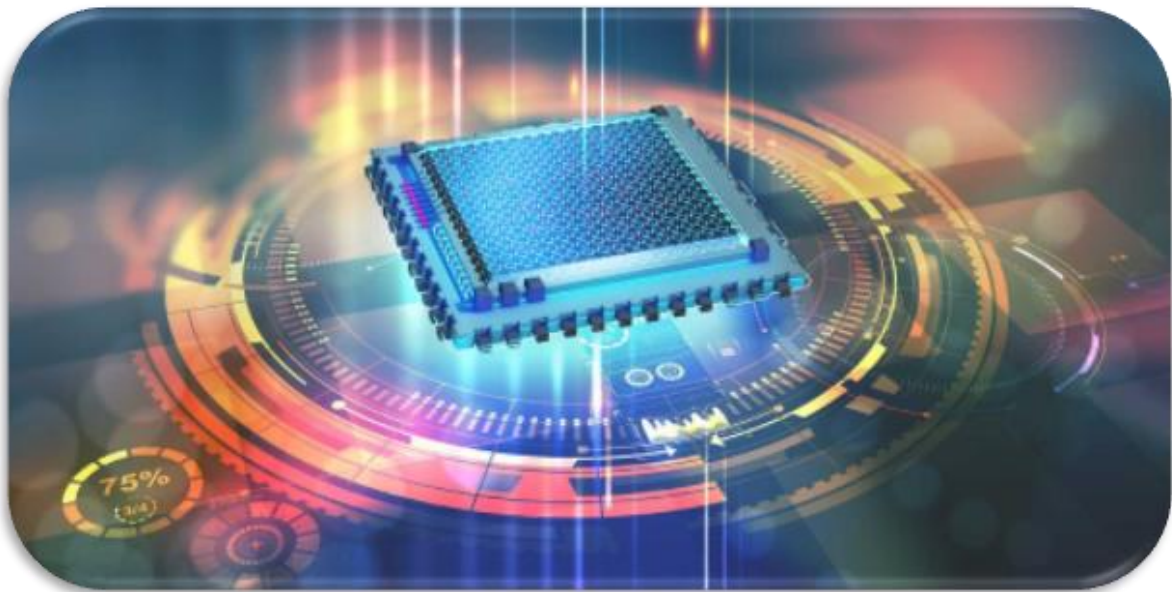


Khalil BAHRI - Xavier BERTAULD – Clément CHAMPION  
Charles DELEFORGE – Axel MARLARD – Julia MOURIER

# Ordinateur Classique Versus Ordinateur Quantique

## Rapport final de projet



Sous la direction de l'enseignant-chercheur Samuel DELEPLANQUE

## Table des matières

Remerciements .....	3
Introduction .....	4
I. Les Calculateurs Quantiques et D-Wave .....	5
1. L'Histoire des Calculateurs Quantiques .....	5
2. D-Wave .....	5
3. Le « Quantum Annealing » .....	6
II. Problème de Max-Cut .....	8
III. Problème de la répartition des projets .....	11
1. Explication du problème .....	11
2. Avec un ordinateur classique .....	12
Étape 1 : Récupération des infos .....	12
Étape 2 : Construction du modèle mathématique .....	13
Étape 3 et 4 : Résoudre et afficher les résultats .....	15
3. Avec un ordinateur quantique .....	16
Étape 1 et 2 : Modification du modèle mathématique et ajout des contraintes .....	16
Étape 3 et 4 : Lancement et résultats du programme .....	22
4. Comparaison des résultats .....	25
IV. Gestion de projet .....	26
Conclusion .....	27
Bibliographie .....	28
Table des annexes .....	29



## Remerciements

Au terme de ce travail, nous tenons à remercier notre responsable de projet et enseignant-chercheur Monsieur Samuel DELEPLANQUE, pour son aide apportée et sa bienveillance tout au long du projet.

De plus, nous souhaitons remercier l'enseignante-chercheuse Madame Isabelle LEFEBVRE pour l'intérêt qu'elle aura porté à notre projet, ainsi que pour la mise en avant de notre travail de recherche sur Twitter.

Également, nous voulons remercier le responsable département Mathématiques & Informatique Gabriel CHÊNEVERT pour ses conseils et son aide sur ce projet.

Et enfin, nous tenons à remercier également notre cliente et enseignante-chercheuse Madame Pascale DIENER, pour nous avoir permis de travailler sur la répartition des projets et pour son aide sur ce problème.

## Introduction

Le sujet « Ordinateur Classique Versus Ordinateur Quantique » s'inscrit dans le cadre des projets du Semestre 2 de Master 1 à l'ISEN Lille. Il a duré 6 semaines du 15 Mars au 20 Avril et a été réalisé par Khalil BAHRI, Xavier BERTAULD, Clément CHAMPION, Charles DELEFORGE, Axel MARLARD et Julia MOURIER sous la direction de l'enseignant-chercheur Samuel DELEPLANQUE. Le projet consiste à étudier les avancées réalisées par les ordinateurs quantiques à base de recuit quantique (Quantum Annealing en anglais) en utilisant la résolution de problèmes d'optimisation et pour cela, nous avons utilisé les ordinateurs quantiques de la société D-Wave.

Le but étant de vérifier si ces ordinateurs quantiques sont plus performants que des ordinateurs classiques en résolvant des problèmes d'optimisation puis en testant en et comparant les différents résultats trouvés.

Pour cela, nous avons décidé de traiter deux problèmes sur ce projet :

- Le problème Max-cut ou appelé problème de coupe maximum
- Le problème de répartition des projets de M1

Les thèmes prédominants étant la recherche dans le domaine quantique et le développement d'algorithme ici en python.

Nous tâcherons d'abord d'introduire les bases de l'ordinateur quantique et du principe de recuit quantique puis dans une seconde partie le problème de Max-Cut, ensuite le problème de répartition des projets en version classique et quantique pour finir par une brève explication de notre gestion de projet.

## I. Les Calculateurs Quantiques et D-Wave

### 1. L'Histoire des Calculateurs Quantiques

Les calculateurs quantiques sont conceptualisés dans les années 80 mais jusque dans les années 90, il existe un doute sur la possibilité de réaliser ces calculateurs quantiques notamment dues au phénomène d'interaction du système quantique avec son environnement, qui provoque la décohérence et la perte de tout ou partie de l'information calculée.

C'est justement dans ces années 90 que les premiers calculateurs apparaissent mais sont rapidement limités dans la réalisation du qubit, essentiel aux calculateurs, le qubit étant l'analogue quantique du bit, basé sur le principe de superposition quantique (un même état quantique peut posséder plusieurs valeurs pour une certaine quantité observable) et d'intrication quantique (lequel deux particules forment un système lié, et présentent des états quantiques dépendant l'un de l'autre).

Ce n'est qu'en 2008 que la production des qubits est simplifiée et en 2009, l'université de Yale met au point le premier QPU (analogue quantique d'un CPU) composé de deux qubits. Ouvrant la voie aux calculateurs quantiques utilisant les portes logiques quantiques.

### 2. D-Wave

Les ordinateurs quantiques que nous avons utilisés lors de notre projet sont ceux de l'entreprise D-Wave, ceux-ci sont conceptualisés sur un principe différent des calculateurs quantiques dit « universels » car ils reposent sur le recuit quantique ou « quantum annealing ». D-Wave développe cette technologie maintenant depuis 2007 avec la création de leur premier prototype composé de 4 qubits. Il commercialise leur premier modèle en 2011 avec un modèle composé de 128 qubits. En fin d'année 2020, D-Wave commercialise leur dernier calculateur « Advantage » composé de 5640 qubits. D-Wave a rencontré le succès dès la commercialisation de leur premier calculateur avec notamment des clients comme la NASA ainsi que la société Lockheed Martin.

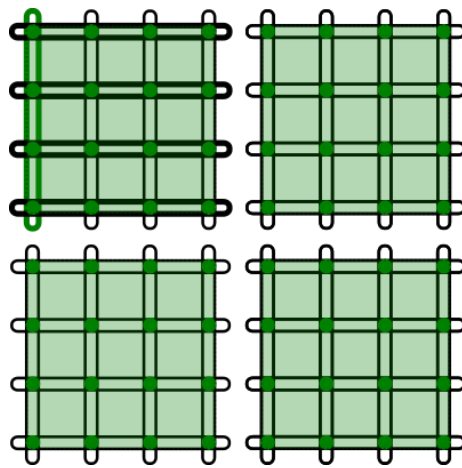
### 3. Le « Quantum Annealing »

Comme cité précédemment, le « quantum annealing » est la technologie qu'utilise D-Wave pour leurs calculateurs quantiques. Leur dernier modèle possède 5640 qubits et une connectivité de 15. La connectivité représente le nombre de connexions entre chaque qubit qu'on peut assimiler à un « chimera graph ».

Ce « chimera graph » se décompose ainsi :

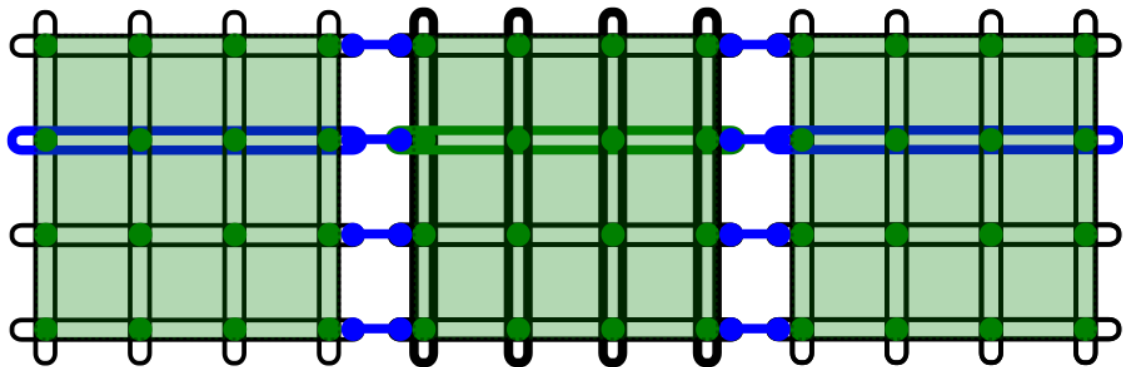
- Coupleur interne :

Les coupleurs internes couplent donc jusqu'à 4 qubits horizontalement.

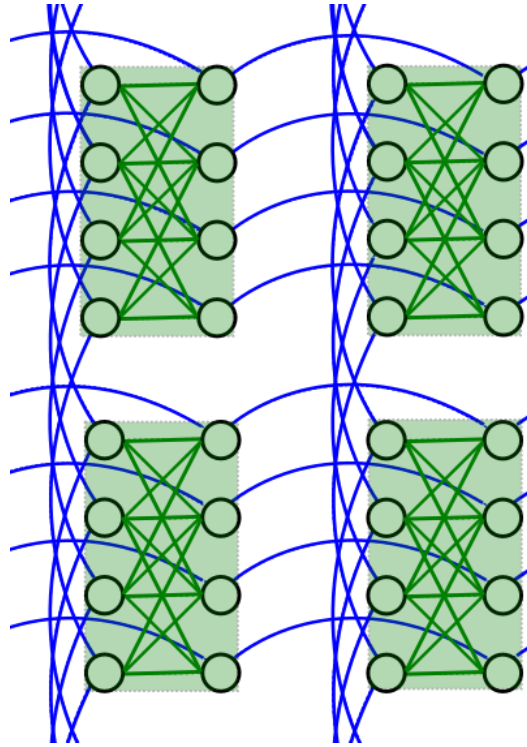


- Coupleur externe :

Les coupleurs externes ici permettent de connecter les qubits colinéaires.



On a donc cette représentation :



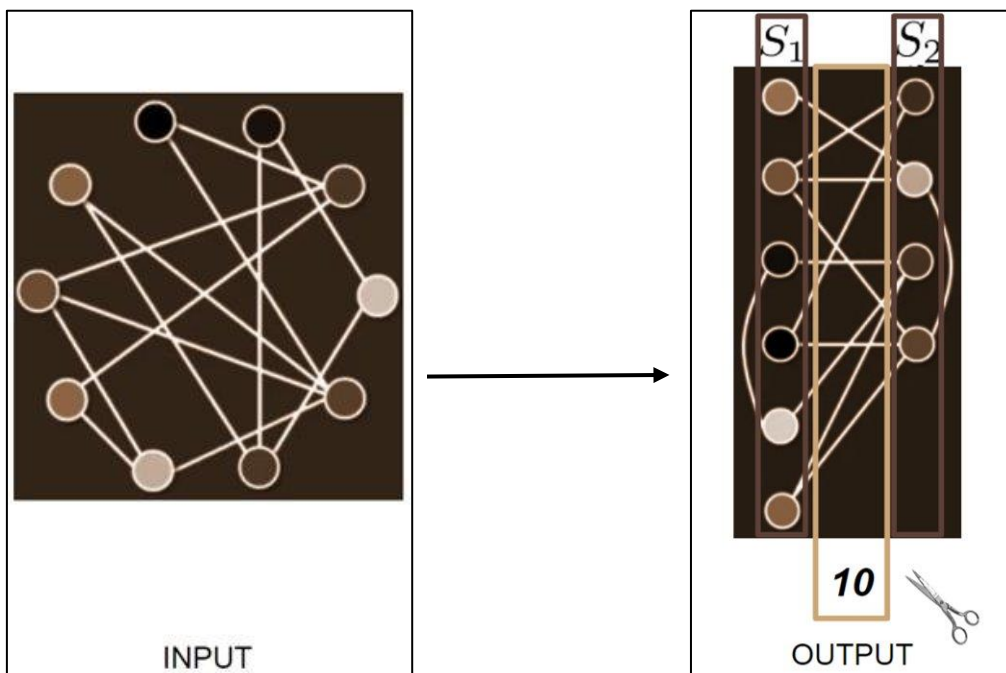
Ici on peut voir que chaque qubit est relié à 6 autre qubits, 4 avec des coupleurs internes (ici en vert) et 2 avec des coupleurs externes (ici en bleu). Evidemment, pour la machine la plus récente, étant donné que la connectivité est de 15 les connections entre les qubits sont différentes.

## II. Problème de Max-Cut

Nous donc été introduits à l'ordinateur quantique par la résolution d'un fameux problème, celui de la coupe maximum (« Max cut problem » en anglais).

En théorie des graphes et en algorithmique, le problème de coupe maximum est le fait, à partir d'un graphique comportant des sommets et des arêtes reliant ces sommets, de séparer les différents sommets en deux ensembles, le but étant d'avoir que le nombre d'arêtes reliant les deux ensembles soit le plus grand possible.

Par exemple :



Le code permettant de résoudre le problème nous ayant été fourni, nous avons tout de suite pu commencer à nous intéresser aux aspects techniques du problème.


Nous avons ainsi voulu commencer par déterminer le nombre maximum de sommets / arêtes que pouvait prendre en charge un ordinateur classique lors de la résolution du problème (qui se fait via Python). Puis nous avons fait de même pour la résolution sur l'ordinateur quantique.

Nous avons donc déterminé que la limite de sommets dépend du nombre moyen d'arêtes associées à chaque sommet. Pour un ordinateur classique avec en moyenne 3 arêtes par sommet, l'ordinateur ne pourra ainsi pas résoudre le problème à partir de 23 sommets, tandis que pour l'ordinateur quantique que nous avons utilisé, la limite est d'un peu plus que 370 sommets (toujours pour 3 arêtes en moyenne). Si on travaille sur un graphique où chaque



sommet possède en moyenne 2 arêtes, il sera possible pour l'ordinateur quantique de résoudre le problème avec + de 500 sommets.

```

1 import numpy as np
2 import dimod
3
4 from dwave.system.samplers import DWaveSampler
5 from dwave.system.composites import EmbeddingComposite
6
7
8 J = 
9 h = {}
10 model = dimod.BinaryQuadraticModel( h, J, 0.0, dimod.SPIN)
11 print("Le model résolu est le suivant:")
12 print(model)
13 print()
14
15
16 sampler = EmbeddingComposite(DWaveSampler(Solver='Advantage_system1.1'))
17 sampler_name = sampler.properties['child_properties']['chip_id']
18 response = sampler.sample(model, num_reads = 50)
19 print("The solution obtained by D-Wave's quantum annealer", sampler_name, "is")
20 print(response)
21 print()

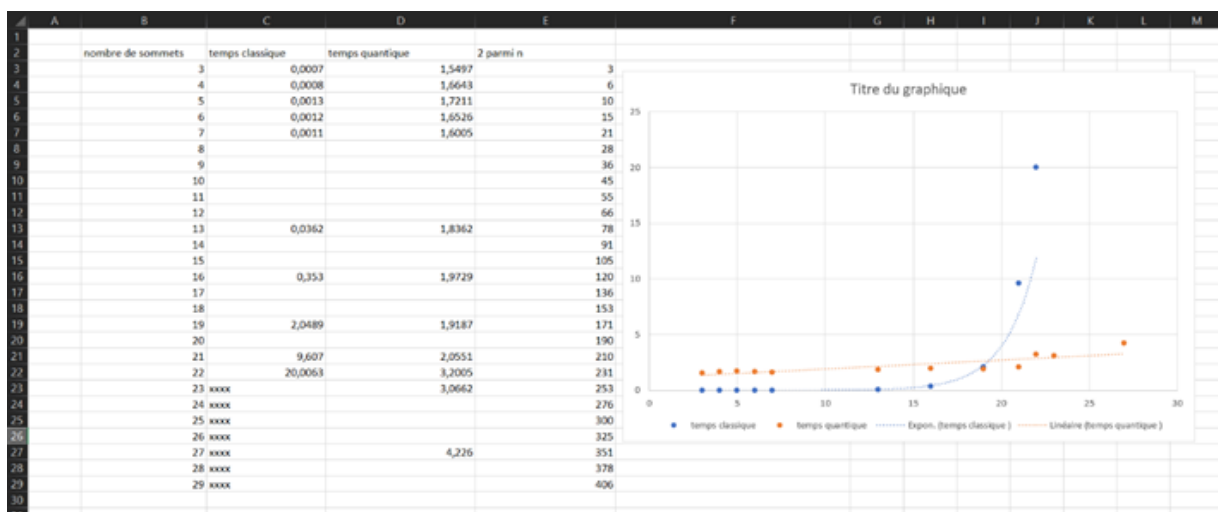
```

Code de résolution du problème en quantique, où J est un graphe généré aléatoirement de X sommets et Y arêtes


Nous avons émis l'hypothèse que la limitation des sommets en quantique pouvait venir du code en lui-même, mais après analyse de toutes les fonctions présentes dans le code, et s'être intéressé à d'autres langages / IDE ( Qiskit ), nous avons déterminé que les limitations provenaient de la machine en elle-même.

Nous avons ensuite comparé la vitesse de calcul entre l'ordinateur classique et l'ordinateur quantique pour le problème de max cut.

Les résultats sont explicités ci-dessous :



On constate que le temps de résolution est proportionnel à la complexité du problème pour l'ordinateur classique quand il est linéaire pour l'ordinateur quantique.



Ce graphique montre donc que l'ordinateur quantique est globalement plus efficace qu'un ordinateur classique quand il s'agit de résoudre un problème d'une grande taille, mais sera cependant moins rapide quand le problème est de taille plus modeste.

Nous avons aussi étudié les probabilités de trouver une solution la plus optimale possible en faisant varier le nombre de calculs ainsi que le nombre de sommets et d'arêtes.

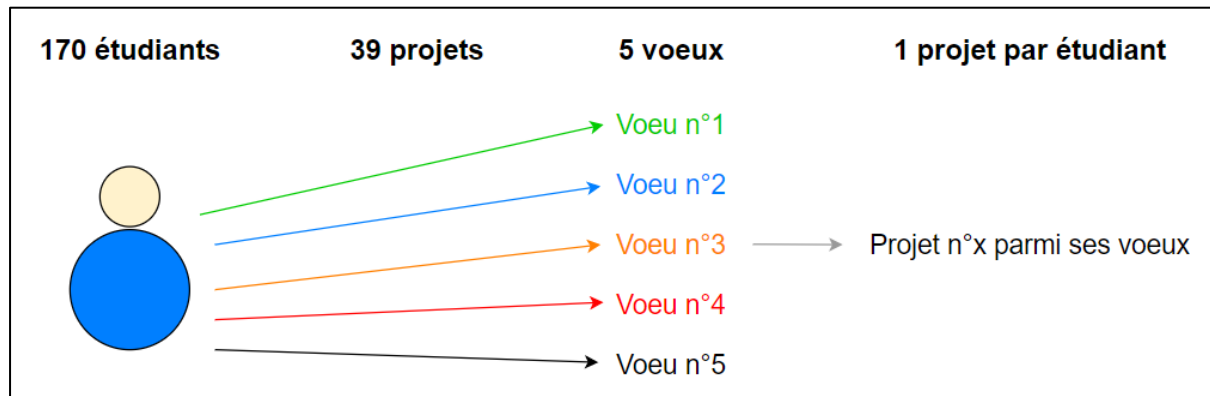
Nous avons constaté que pour le même nombre de sommets et d'arêtes (15 et 30), le nombre de calcul n'a aucune influence sur les probabilités de trouver une solution optimale.

Nous avons voulu, pour conclure sur ce problème, avoir par la suite la réponse à certaines questions que nous nous posions concernant l'ordinateur quantique et les méthodes de résolution proposées. Nous avons donc pris contact avec D-Wave et les échanges que nous avons eu sont disponibles en annexe. (Annexe 1)

### III. Problème de la répartition des projets

#### 1. Explication du problème

La répartition des projets de M1 peut se résumer de cette manière :



Nous avons donc 170 étudiants qui doivent choisir 5 projets et les classer par ordre de préférence, ce qui fait au total 850 vœux. Et sur ces vœux, il faut en garder 170 pour répartir chaque étudiant à un seul projet en essayant de les satisfaire au maximum. Mais cela mène à plusieurs problèmes.

Actuellement, notre cliente Pascale DIENER répartit les projets à la main et cela lui prend environ 8 heures. De plus, avec cette méthode, elle arrive à un taux d'étudiants ayant leur vœu 1 de 58.8%.

Nos objectifs sont donc :

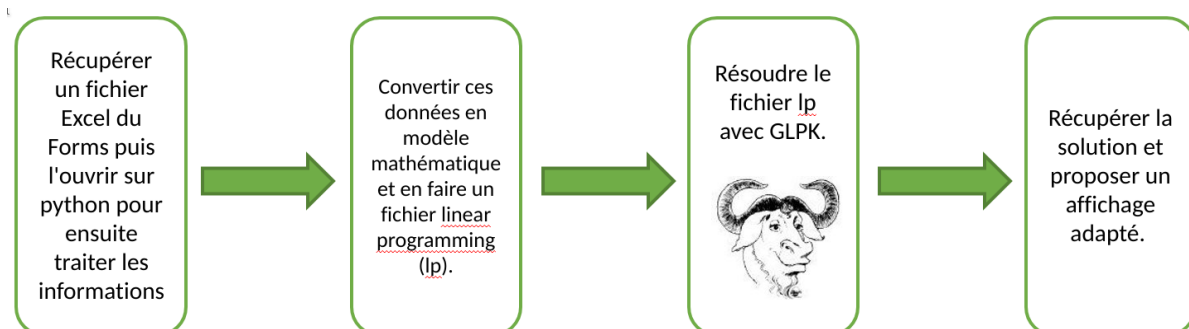
- Créer un algorithme permettant d'éviter la répartition à la main.
- Augmenter le taux d'étudiants ayant leur vœu 1.
- Augmenter le taux de satisfaction général c'est-à-dire minimiser le taux d'élèves ayant le vœu 4 ou 5.
- Respecter le cahier des charges fixé par notre cliente.

## 2. Avec un ordinateur classique

Pour résoudre le problème de répartition des projets nous allons devoir passer par plusieurs étapes que nous avons planifiées avant de commencer. Toutes ces étapes sont contenues dans un programme python qu'il suffit de lancer pour résoudre le problème. Nous allons donc par commencer résumer les différentes étapes puis nous rentrerons dans le détail de certaines d'entre elles et enfin nous analyserons rapidement les résultats (cette analyse sera plus détaillée plus tard lorsque nous aurons les résultats du quantique pour comparer).

La première étape est de récupérer le tableur résultant des différentes réponses des élèves. Ce tableur nous l'ouvrons avec python afin d'en récupérer les informations. La deuxième étape sera de construire un modèle mathématique à partir de ses données, nous détaillerons comment faire légèrement plus bas. Puis il faut convertir ce modèle mathématique afin qu'un solveur puisse le comprendre et le résoudre. Pour cela le programme python crée un fichier dit linear programming (abréviation lp). Ce fichier lp sera ensuite envoyé à GLPK (GNU Linear Programming Kit) un logiciel libre qui permet de résoudre des fichiers LP. Enfin nous avons récupéré la solution que nous donne GLPK toujours dans python et proposons un affichage adapté.

Voici le résumé des étapes :



*Schéma résumé des étapes*

### Étape 1 : Récupération des infos

Afin de d'ouvrir le fichier Excel dans python, nous utilisons le package pandas qui permet de récupérer une case de tableur à l'aide de ses coordonnées (i,j). Toutes les informations sont ensuite mises dans des listes contenant par exemple des éléments de classe *Student* contenant dans ce cas des informations comme le nom de l'étudiant, son adresse mail, ses choix... C'est également à ce moment qu'une fonction permet de vérifier si les étudiants ont correctement rempli le formulaire, si ce n'est pas le cas un message à la fin affiche le nom de tous les étudiants n'ayant pas bien rempli le questionnaire. Lorsque cette étape est terminée, nous allons ensuite appeler les fonctions qui permettent de construire le modèle mathématique en vue de la résolution avec GLPK.

## Etape 2 : Construction du modèle mathématique

Le problème était le suivant : comment traduire notre problème qui semble concret en un modèle mathématique générique qui pourra être réutilisé chaque année ?

Pour cela, le modèle se base sur une fonction objectif dont voici la forme :

$$Obj = \min \sum x_{ij} w_{ij}$$

*w<sub>ij</sub> : weight associated with the student's i option for projet j*

Il s'agit d'une somme à minimiser, les composantes de cette somme sont des produits à 2 facteurs. Le premier est  $x_{ij}$ ,  $x_i$  représente un étudiant (  $x_0$  est l'étudiant 0 ,  $x_1$  l'étudiant 1 etc) et  $j$  représente de la même manière un numéro de projet. Ainsi  $x_{ij}$  vaut 1 si l'étudiant  $i$  a le projet  $j$  d'attribué et 0 si l'étudiant n'obtient pas le projet  $j$ .

La deuxième composante de ce produit  $w_{ij}$  représente le poids du choix de l'étudiant  $i$  pour le projet  $j$ . Plus le choix est prioritaire pour l'étudiant plus  $w_{ij}$  est petit, ainsi le solveur aura tendance à choisir un projet prioritaire pour l'étudiant afin de minimiser la somme.

Mais jusque-là cela ne résout pas notre problème, pour cela nous allons devoir ajouter des contraintes. La première sert à ce que le nombre d'étudiants par projet ne dépasse pas le nombre de places disponibles. Pour cela, la somme des élèves dans un projet doit être inférieure au nombre de places, voici comment cela se traduit :

$$\forall \text{ project } j$$
$$\sum_i^{n\_students} x_{ij} \leq room_{max}$$

Nous ajoutons également une contrainte pour que chaque étudiant ait exactement 1 projet, pour chaque étudiant la somme des projets doit donc être égale à 1 :

$$\forall \text{ student } i$$
$$\sum_j^{n\_projects} x_{ij} = 1$$

Enfin nous avons dû ajouter une dernière contrainte afin qu'un projet ait au minimum 3 étudiants. Afin d'ajouter cette contrainte à notre fichier lp, nous l'avons également traduit mathématiquement :

$$\begin{aligned} & \forall \text{ project } j \\ & \sum_i^n x_{ij} \leq M \cdot \text{activeProject}_j \\ & \forall \text{ project } j \\ & \sum_i^n x_{ij} \geq 3 \cdot \text{activeProject}_j \\ & \text{activeProject}_j \in \{0, 1\} \text{ and } M = \text{room}_{\max j} \end{aligned}$$

On voit apparaître ici une contrainte de type "big M". Celle-ci permet de forcer le nombre d'étudiant à 0 lorsque activeProject vaut 0 mais permet lorsque le nombre d'étudiant est supérieur à 3 d'être le nombre d'étudiants souhaité pour le projet.

En général, M est fixé arbitrairement comme une valeur très grande. Cependant, plus M est petit, plus le calcul est rapide donc il faut choisir M de manière réfléchi si on veut le temps de calcul le plus petit possible. Ainsi M peut être choisi comme étant le nombre de places dans un projet. On peut alors remplacer la contrainte précédente limitant le nombre de places.

Maintenant que nous avons préparé notre modèle mathématique, nous allons pouvoir créer notre fichier de linear programming à partir de celui-ci. Voici à quoi un fichier lp ressemble au final, ici il s'agit d'un exemple avec 9 étudiants et 3 projets :

```
Minimize
obj: + 10000 x0.0 + 2 x0.1 - 1 x0.2 + 10000 x1.0 - 1 x1.1 + 5 x1.2 + 10000 x2.0 - 1 x2.1 + 5 x2.2
x7.1 + 5 x7.2 - 1 x8.0 + 10000 x8.1 + 5 x8.2
Subject To
c0: + x0.0 + x1.0 + x2.0 + x3.0 + x4.0 + x5.0 + x6.0 + x7.0 + x8.0 - 3 y0 >= 0
c1: + x0.0 + x1.0 + x2.0 + x3.0 + x4.0 + x5.0 + x6.0 + x7.0 + x8.0 - 3 y0 <= 0
c2: + x0.1 + x1.1 + x2.1 + x3.1 + x4.1 + x5.1 + x6.1 + x7.1 + x8.1 - 3 y1 >= 0
c3: + x0.1 + x1.1 + x2.1 + x3.1 + x4.1 + x5.1 + x6.1 + x7.1 + x8.1 - 4 y1 <= 0
c4: + x0.2 + x1.2 + x2.2 + x3.2 + x4.2 + x5.2 + x6.2 + x7.2 + x8.2 - 2 y2 <= 0
c5: + x0.0 + x0.1 + x0.2 = 1
c6: + x1.0 + x1.1 + x1.2 = 1
c7: + x2.0 + x2.1 + x2.2 = 1
c8: + x3.0 + x3.1 + x3.2 = 1
c9: + x4.0 + x4.1 + x4.2 = 1
c10: + x5.0 + x5.1 + x5.2 = 1
c11: + x6.0 + x6.1 + x6.2 = 1
c12: + x7.0 + x7.1 + x7.2 = 1
c13: + x8.0 + x8.1 + x8.2 = 1
Binary
x0.0
x0.1
x0.2
x1.0
x1.1
x1.2
x2.0
x2.1
x2.2
x3.0
x3.1
x3.2
x4.0
x4.1
x4.2
x5.0
x5.1
x5.2
x6.0
x6.1
x6.2
x7.0
x7.1
x7.2
x8.0
x8.1
x8.2
y0
y1
y2
End
```

« Minimize » correspond donc à notre fonction objectif. La catégorie « Subject To » contient toutes les contraintes auxquelles le modèle est soumis. Enfin la catégorie « Binary » permet de déclarer que toutes les variables qui sont en dessous sont binaires.

### Etape 3 et 4 : Résoudre et afficher les résultats

Une fois ce fichier créé, il est envoyé par python à GLPK qui se charge de le résoudre. On entre alors dans la dernière étape de notre programme python. Les résultats de GLPK sont récupérés et sont affichés de la manière suivante :

```
Pour le projet 25 (4/4)
étudiant : 44 => charles.sailliot@isen.yncrea.fr
étudiant : 58 => kevin.ficet@isen.yncrea.fr
étudiant : 70 => camille.benoist@student.yncrea.fr
étudiant : 107 => geoffrey.delfosse@isen.yncrea.fr

Pour le projet 26 (5/6)
étudiant : 1 => thomas.candusso@isen.yncrea.fr
étudiant : 9 => baptiste.royer@isen.yncrea.fr
étudiant : 67 => benjamin.maerten@isen.yncrea.fr
étudiant : 134 => louis-henri.haguet@student.yncrea.fr
étudiant : 143 => theodore.westeel@isen.yncrea.fr
```

Cependant plusieurs options sont proposées à l'utilisateur. Celui-ci peut-choisir d'afficher uniquement le nombre d'élèves par projet, demander de n'afficher qu'un projet spécifique etc.

Ceci clôture la résolution à l'aide d'un ordinateur classique. Une notice pour utiliser notre programme est disponible (Annexe 2). Nous allons maintenant procéder à la résolution en quantique.

### 3. Avec un ordinateur quantique

Puisque l'objectif final de ce projet est de comparer les ordinateurs classiques aux ordinateurs quantiques, nous sommes ensuite passés à une résolution sur les ordinateurs de D-Wave.

#### Etape 1 et 2 : Modification du modèle mathématique et ajout des contraintes

Pour cela, nous avons dû commencer par redéfinir le problème mathématique puisque le QUBO nécessite que les contraintes soient intégrées à la fonction objectif. Autrement dit, il faut incorporer les différentes contraintes dans la fonction objectif.

Pour cela, il faut utiliser un multiplicateur de Lagrange de la façon suivante :

$$\text{objective\_function} + \gamma(\text{contrainte})$$

Où  $\gamma$  est le multiplicateur de Lagrange permet la relaxation d'une contrainte. Ce paramètre est un scalaire permettant de donner plus ou moins de la valeur à une contrainte. Pour trouver sa valeur, il faudra réaliser empiriquement plusieurs tests.

Pour la suite, replaçons-nous dans l'exemple précédent. Voici ce que nous avons :

Modèle mathématiques pour 9 élèves pour 3 projets de 3, 4 et 2 places :

$x_{ij}$  est défini tel que  $x_{ij} = 1$  si l'élève  $i$  obtient le projet  $j$ , 0 sinon.

Contraintes :

Place par projet :

$$\sum x_{i0} \leq 3$$

$$\sum x_{i1} \leq 4$$

$$\sum x_{i2} \leq 2$$

1 seul Projet par étudiant :

$$\sum x_{0j} = 1$$

$$\sum x_{1j} = 1$$

$$\sum x_{2j} = 1$$

...

$$\sum x_{8j} = 1$$

Nous avons alors deux types de contraintes :

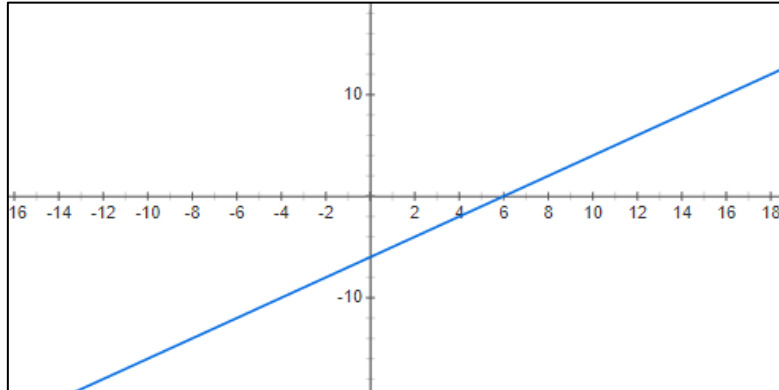
- Celle qui donne le nombre d'étudiants par projet
- Celle qui oblige un étudiant à avoir un unique projet



La première contrainte se traduit par une inégalité pour chaque projet. Ainsi, elle peut être traduite par la somme suivante :

$$\begin{array}{c} \forall \text{ projet } j \\ \gamma_j \left( \sum_i^9 x_{ij} - n_{room\ j} \right) \\ n_{room} : \text{nombre de places pour le projet } j \end{array}$$

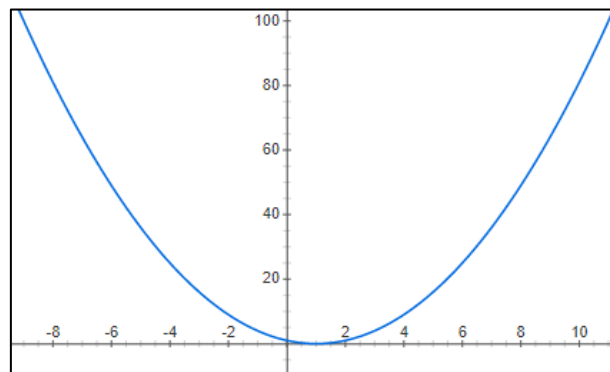
La fonction  $y = x_{ij} - n_{room}$  peut se tracer de la façon suivante (où  $n_{room} = 6$ ), forçant ainsi la somme à respecter la contrainte :



Pour la deuxième sorte de contrainte, on a une égalité stricte. Ainsi, il est nécessaire de mettre la somme au carré.

$$\begin{array}{c} \forall \text{ étudiant } i \\ \gamma_{(3+i)} \left( \sum_j^3 x_{ij} - 1 \right)^2 \end{array}$$

En effet, cela correspond à une parabole dont le minimum est situé en 1 :



Avec ces deux types de contraintes on obtient le résultat suivant :

$$\begin{array}{l} \min \left( \sum_i^9 \sum_j^3 x_{ij} w_{ij} \right) + \gamma_0 \left( \sum_i^9 x_{i0} - 3 \right) + \gamma_1 \left( \sum_i^9 x_{i1} - 4 \right) + \gamma_2 \left( \sum_i^9 x_{i2} - 2 \right) + \\ \gamma_3 \left( \sum_j^3 x_{0j} - 1 \right)^2 + \gamma_4 \left( \sum_j^3 x_{1j} - 1 \right)^2 + \gamma_5 \left( \sum_j^3 x_{2j} - 1 \right)^2 + \dots + \gamma_{11} \left( \sum_j^3 x_{8j} - 1 \right)^2 \end{array}$$

Puisque les contraintes indiquant qu'un étudiant ne peut avoir qu'un seul projet semblent avoir la même importance, on peut supposer :

$$\{\gamma_3 = \gamma_4 = \dots = \gamma_{11} = \lambda_2\}$$

De plus, on peut simplifier le calcul avec la formule :

$$((\sum_{i=0}^n x_i) - C)^2 = \sum_{i=0}^n x_i^2 + 2 \sum_{i=0}^n \sum_{j>i}^n x_i x_j - 2C \sum_{i=0}^n x_i + C^2$$

On peut remarquer également que  $x_{ij}$  est un booléen, et donc que  $x_{ij}^2 = x_{ij}$ .

Ce qui nous donne :

$$\begin{aligned} \min(&\sum x_{ij} w_{ij}) + (\gamma_0 \sum_i x_{i0} + \gamma_1 \sum_i x_{i1} + \gamma_2 \sum_i x_{i2}) + \lambda_2 (2 \sum_j \sum_{k=j+1}^3 x_{0j} x_{0k} - \sum_{j=0}^3 x_{0j} \\ &+ 2 \sum_j \sum_{k=j+1}^3 x_{1j} x_{1k} - \sum_{j=0}^3 x_{1j} \dots + 2 \sum_j \sum_{k=j+1}^3 x_{8j} x_{8k} - \sum_{j=0}^3 x_{8j}) - 3\gamma_0 - 4\gamma_1 - 2\gamma_2 + 9\lambda_2 \end{aligned}$$

Avec ce modèle, il a été facile de généraliser la formule :

$$\begin{aligned} &\min(\sum x_{ij} w_{ij}) \\ &\forall \text{ project } j : + \gamma_j \sum_j^{n_{\text{étudiants}}} x_{ij} \\ &\forall \text{ student } i : + \lambda_2 (2 \sum_j^{n_{\text{projets}}} \sum_{k=j+1}^{n_{\text{projets}}} x_{ij} x_{ik} - \sum_{j=i}^{n_{\text{projets}}} x_{ij}) \end{aligned}$$

Les constantes sont volontairement négligées car elles n'interviennent pas dans le code puisqu'elles permettent seulement de donner un offset à la somme.

Avec ceci on peut passer à la programmation du code python dans l'IDE de D-wave. Pour cela il est nécessaire de remplir les coefficients précédents dans une matrice. Nous la noterons Q par la suite.

Nos variables  $x_{ij}$  sont à deux dimensions, nous avons donc eu recours à un petit stratagème pour les mettre dans une seule dimension. La fonction « get\_index », inspirée d'un exemple de D-Wave permet de faire ce changement de dimension. A partir du couple (index\_étudiant, index\_projet) il ressort un unique index,  $i = \text{index\_étudiant} * n_{\text{projets}} + \text{index\_projet}$ .

```
def get_index(student_index, project_index, n_projects):
    """
    Get the index of the matrix corresponding to the index of students and projects
    (opposite of get_student_and_project)

    :param student_index: index of the student
    :param project_index: index of the project
    :param n_projects: number of projects

    :type student_index: integer
    :type project_index: integer
    :type n_projects: integer

    :return index: index corresponding to project_student
    :rtype students: integer
    """
    return student_index * n_projects + project_index
```

Avec cette fonction et notre modèle mathématique on peut donc remplir la matrice Q.

- Pour la première contrainte, on sépare les deux sommes :

$$\forall \text{ student } i : +\lambda_2 \left( 2 \sum_j^{n_{\text{projects}}} \sum_{k=j+1}^{n_{\text{projects}}} x_{ij} x_{ik} - \sum_{j=i}^{n_{\text{projects}}} x_{ij} \right)$$

```
# Constraint 1 : One project per student
for student_index in students:
    for project_index in projects:
        ind1 = get_index(student_index, project_index, len(projects))
        Q[(ind1, ind1)] -= lagrange_parameter_only_one

for student_index in students:
    for project_index in projects:
        for project_index_2 in projects:
            ind1 = get_index(student_index, project_index, len(projects))
            ind2 = get_index(student_index, project_index_2, len(projects))
            Q[(ind1, ind2)] += lagrange_parameter_only_one
```

Pour le multiplicateur de Lagrange, il a été choisi empiriquement après plusieurs essais. Il a été fixé à 100000 en partie à cause des poids des options fixés à -10000, -5000, -1000, -500, -10 (de l'option 1 à l'option 5). Lorsque l'étudiant n'a pas choisi un projet en option, son poids est placé à 100000 pour empêcher l'ordinateur de sélectionner un projet non choisi par l'étudiant.

- Pour la deuxième contrainte :

$$\forall \text{ project } j : +\gamma_j \sum_i^{n_{\text{students}}} x_{ij}$$

Dans un premier temps, nous avons essayé avec le modèle décrit plus haut, il correspond aux parties commentées :

```
# Constraint 2 : Room per projects => lines 109 and 118 decomment and comment lines 110 & 119
for project_index in projects:
    for student_index in students:
        ind1 = get_index(student_index, project_index, len(projects))
        # Q[(ind1, ind1)] += lagrange_parameter_room[project_index]
        Q[(ind1, ind1)] -= (2*room[project_index])*lagrange_parameter_room[project_index]

for project_index in projects:
    for student_index in range(len(students)):
        for student_index_2 in range(student_index, len(students)):
            ind1 = get_index(student_index, project_index, len(projects))
            ind2 = get_index(student_index_2, project_index, len(projects))
            # Q[(ind1, ind2)] += 0
            Q[(ind1, ind2)] += 2*lagrange_parameter_room[project_index]
```

Cependant nous avons remarqué que les contraintes du nombre de places n'étaient pas suffisamment bien respectées. Nous avons donc décidé de changer légèrement le modèle pour passer encore une fois à une égalité stricte. Cela fonctionne bien car il y a plus de places proposées que d'étudiants, il est donc quasiment impossible de dépasser le nombre de places. Ce qui se traduit mathématiquement par :

$$\forall \text{ project } j : +\gamma_j \left( -(2 \times n_{\text{room}} - 1) \sum_j^{n_{\text{students}}} x_{ij} + 2 \sum_j^{n_{\text{projects}}} \sum_{k=j+1}^{n_{\text{projects}}} x_{ij} x_{ik} \right)$$

De plus, nous avons choisi de favoriser les projets où il y a le plus de places proposées en paramétrant le multiplicateur de Lagrange de la façon suivante :

```
# coefficient forcing the project to have the number of students needed (helping more the project with more students)
coeff = 2000
roomCoeffed = roomWithCoeff(room, coeff)
```

La fonction « roomWithCoeff » multiplie chaque élément de room par le coefficient (ici 2000).

Une fois que la matrice est correctement remplie, on peut l'envoyer à l'ordinateur. Nous avons laissé la possibilité à l'utilisateur d'envoyer le problème sur un ordinateur quantique en l'occurrence *Advantage1.1* ou l'ordinateur hybride *hybrid\_binary\_quadratic\_model\_version2*.

Dans chacun des cas, la matrice a besoin d'être intégrée sur l'ordinateur et pour cela une ligne de code supplémentaire est nécessaire. Celle-ci est encadrée en rouge sur le screen ci-dessous.

```

# Choice of computer
print("\nChoisissez Ordi hybride (h) ou Ordi quantique (q) : ")
ordi = input()

time0 = time.time()

if ordi == 'Q' or ordi == 'q':
    print("You chose the quantum D-wave computer\nSending to the d-wave computer...")
    sampler = EmbeddingComposite(DWaveSampler())
    time1 = time.time()
    results = sampler.sample_qubo(Q,num_reads=5000)
else :
    print("You chose the hybrid D-wave computer\nSending to the d-wave computer...")
    bqm = BinaryQuadraticModel.from_qubo(Q, offset=0)
    sampler = LeapHybridSampler()
    time1 = time.time()
    results = sampler.sample(bqm, label='Example - Nurse Scheduling')

```

Dans le cas du problème de cette année, le problème semble trop gros et n'arrive pas à s'intégrer sur l'ordinateur :

```

Sending to the d-wave computer...
Traceback (most recent call last):
  File "/workspace/nurse-scheduling/hybridv2.py", line 426, in <module>
    results = sampler.sample_qubo(Q,num_reads=5000)
  File "/usr/local/lib/python3.7/site-packages/dimod/core/sampler.py", line 240, in sample_qubo
    return self.sample(bqm, **parameters)
  File "/usr/local/lib/python3.7/site-packages/dwave/system/composites/embedding.py", line 238, in sample
    **embedding_parameters)
  File "/usr/local/lib/python3.7/site-packages/minorminer/minorminer.py", line 58, in find_embedding
    suspend_chains=suspend_chains,
  File "minorminer/_minorminer.pyx", line 227, in minorminer._minorminer.find_embedding
RuntimeError: Failed during initialization. This typically occurs when the source graph is unreasonably large or when the embedding problem is over-constrained (via max_fill, initial_chains, fixed_chains, and/or restrict_chains).
Leap IDE /workspace/nurse-scheduling $ 

```

Nous avons tout de même souhaité conserver cette possibilité pour un autre problème.

Une fois l'intégration de la matrice faite, on peut récupérer les résultats en envoyant le modèle à l'ordinateur concerné.

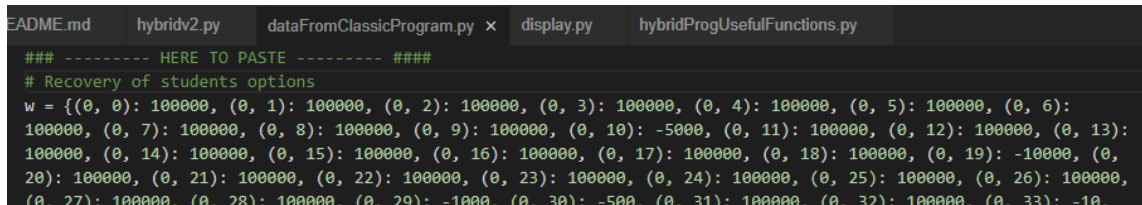
Après un léger traitement du résultat, nous avons repris l'interface codée dans le programme classique. Le seul problème dans le programme est la récupération des informations du fichier XLS. En effet, puisque la lecture du fichier XLS nécessite une librairie externe, il n'est pas possible de le faire dans l'IDE Leap. Ainsi, nous avons ajouté une fonctionnalité dans le programme classique permettant d'exporter les données extraites pour le programme quantique.

```

ImportError: Missing optional dependency 'xlrd'.

```

Ainsi le contenu du fichier finissant par « toTheQuantumProgram.txt » doit être ajouté dans l'IDE dans le fichier « dataFromClassicProgram.py ». Les trois variables w, studentsID et room sont réciproquement les options des étudiants, la liste des emails des étudiants et le nombre de place par projet.



```

### ----- HERE TO PASTE ----- ###
# Recovery of students options
w = {(0, 0): 100000, (0, 1): 100000, (0, 2): 100000, (0, 3): 100000, (0, 4): 100000, (0, 5): 100000, (0, 6): 100000, (0, 7): 100000, (0, 8): 100000, (0, 9): 100000, (0, 10): -5000, (0, 11): 100000, (0, 12): 100000, (0, 13): 100000, (0, 14): 100000, (0, 15): 100000, (0, 16): 100000, (0, 17): 100000, (0, 18): 100000, (0, 19): -10000, (0, 20): 100000, (0, 21): 100000, (0, 22): 100000, (0, 23): 100000, (0, 24): 100000, (0, 25): 100000, (0, 26): 100000, (0, 27): 100000, (0, 28): 100000, (0, 29): -1000, (0, 30): -500, (0, 31): 100000, (0, 32): 100000, (0, 33): -10

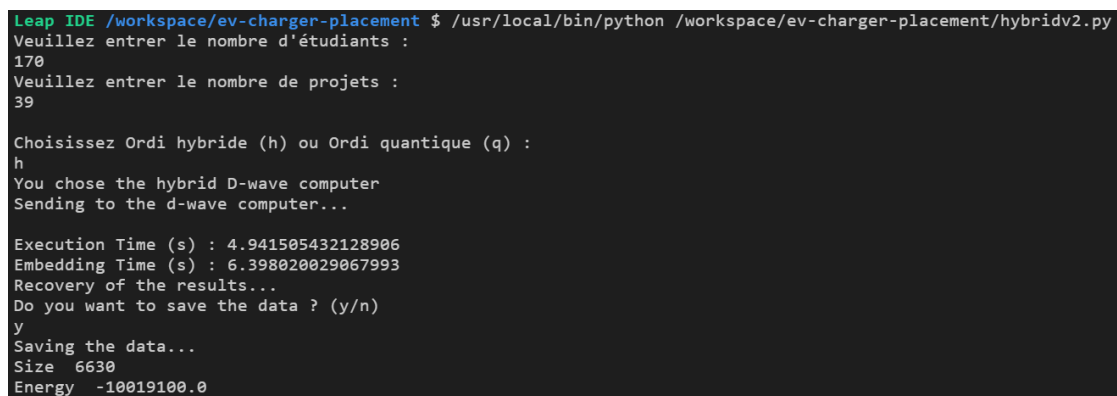
```

Le reste des fonctions utilisées pour ce programme sont assez similaire au programme classique, je vous invite néanmoins à regarder la documentation de ce programme si vous êtes intéressé.

Enfin, si vous souhaitez utiliser notre programme, vous trouverez une notice d'utilisation en Annexe 3.

### Etape 3 et 4 : Lancement et résultats du programme

Voici à quoi ressemble le programme une fois lancé :



```

Leap IDE /workspace/ev-charger-placement $ /usr/local/bin/python /workspace/ev-charger-placement/hybridv2.py
Veuillez entrer le nombre d'étudiants :
170
Veuillez entrer le nombre de projets :
39
Choisissez Ordi hybride (h) ou Ordi quantique (q) :
h
You chose the hybrid D-wave computer
Sending to the d-wave computer...

Execution Time (s) : 4.941505432128906
Embedding Time (s) : 6.398020029067993
Recovery of the results...
Do you want to save the data ? (y/n)
y
Saving the data...
Size 6630
Energy -10019100.0

```

Après avoir renseigné les différentes informations, vous pouvez choisir l'ordinateur sur lequel faire tourner le programme.

On peut observer sur ce screen les différents temps d'exécution et d'intégration. On peut noter que pour ce problème, le temps d'exécution classique est inférieur à celui quantique.

Une autre valeur intéressante ici est l'Energy, elle permet de donner un ordre d'idée du taux de satisfaction, puisque cette valeur correspond au résultat de notre fonction objectif relaxée. Cette valeur et celle que l'on aurait pu décaler avec les constantes « oubliées » de la formule mathématique.

Dans la suite du programme, on retrouve la même interface que pour le programme classique, avec les mêmes possibilités.

Pour le nombre d'étudiants par projet :

```
Si vous voulez voir le nombre d'étudiants pour chaque projet, appuyez sur A
Si vous voulez voir le nom des étudiants pour tous les projets, appuyez sur Z
Si vous voulez voir le nombre d'étudiants pour un projet particulier, appuyez sur E
Si vous voulez voir le taux de satisfaction, appuyez sur R
Si vous voulez quitter, appuyez sur Q
A
Pour le projet 0 il y a 3 étudiants pour 4 places prévues
Pour le projet 1 il y a 5 étudiants pour 5 places prévues
Pour le projet 2 il y n'y a pas d'étudiant pour 2 places prévues
Pour le projet 3 il y a 4 étudiants pour 4 places prévues
Pour le projet 4 il y a 5 étudiants pour 5 places prévues
```

Pour la répartition des étudiants :

```
Z
Pour le projet 0 (3/4)
  étudiant : 92 => anthony.monvoisin@isen.yncrea.fr
  étudiant : 93 => martin.valet@isen.yncrea.fr
  étudiant : 98 => victorien.roussel@isen.yncrea.fr
Pour le projet 1 (5/5)
  étudiant : 19 => thibaut.blasse@isen.yncrea.fr
  étudiant : 52 => joshua.ofori@student.yncrea.fr
  étudiant : 120 => theo.ruchot@isen.yncrea.fr
  étudiant : 125 => nicolas.defoort@isen.yncrea.fr
  étudiant : 153 => lilian.houplin@isen.yncrea.fr
Pour le projet 2 (1/2)
  étudiant : 51 => julien.van-miegem@student.yncrea.fr
Pour le projet 3 (4/4)
  étudiant : 63 => louis.willems@isen.yncrea.fr
```

Pour voir seulement un projet :

```
E
Veuillez entrer le numéro du projet pour lequel vous voulez la liste des étudiants
18
Pour le projet 18 il y a 6 étudiants
  étudiant : 20 => clement.champion@isen.yncrea.fr
  étudiant : 86 => axel.marlard@student.yncrea.fr
  étudiant : 99 => julia.mourier@isen.yncrea.fr
  étudiant : 121 => charles.deleforge@isen.yncrea.fr
  étudiant : 130 => nicolas.bloch@student.yncrea.fr
  étudiant : 165 => khalil.bahri@isen.yncrea.fr
```

Le taux de satisfaction final :

```
R
Pour l'option 1 : 68.82352941176471%
Pour l'option 2 : 15.294117647058824%
Pour l'option 3 : 4.117647058823529%
Pour l'option 4 : 4.705882352941177%
Pour l'option 5 : 5.88235294117647%
```

Maintenant que nous avons des résultats, concentrons-nous sur ceux-ci. Grâce à leur fonctionnement bien particulier, les ordinateurs D-wave nous fournissent une solution différente à chaque compilation. Ainsi, Mme Diener (ou toute personne chargée de la répartition des projets) pourra relancer le programme autant de fois qu'elle le souhaite pour obtenir à chaque fois une répartition différente.

Voici deux solutions et leur taux de satisfaction pour la répartition de notre projet :

```
Pour le projet 18 (6/6)
étudiant : 20 => clement.champion@isen.yncrea.fr
étudiant : 78 => corentin.blondeau@student.yncrea.fr
étudiant : 86 => axel.marlard@student.yncrea.fr
étudiant : 91 => yunchan.guo@student.yncrea.fr
étudiant : 121 => charles.deleforge@isen.yncrea.fr
étudiant : 165 => khalil.bahri@isen.yncrea.fr
```

```
Pour l'option 1 : 68.23529411764706%
Pour l'option 2 : 14.705882352941178%
Pour l'option 3 : 3.5294117647058822%
Pour l'option 4 : 3.5294117647058822%
Pour l'option 5 : 9.411764705882353%
```

Deuxième solution :

```
Pour le projet 18 il y a 6 étudiants
étudiant : 20 => clement.champion@isen.yncrea.fr
étudiant : 86 => axel.marlard@student.yncrea.fr
étudiant : 99 => julia.mourier@isen.yncrea.fr
étudiant : 121 => charles.deleforge@isen.yncrea.fr
étudiant : 130 => nicolas.bloch@student.yncrea.fr
étudiant : 165 => khalil.bahri@isen.yncrea.fr
```

```
Pour l'option 1 : 69.41176470588235%
Pour l'option 2 : 12.352941176470589%
Pour l'option 3 : 5.294117647058823%
Pour l'option 4 : 4.705882352941177%
Pour l'option 5 : 8.235294117647058%
```



On peut même arriver à un pourcentage d'option 1 supérieur à la solution classique, mais qui implique un pourcentage d'option 3, 4 ou 5 supérieurs :

```
Pour l'option 1 : 70.0%
Pour l'option 2 : 11.176470588235295%
Pour l'option 3 : 6.470588235294119%
Pour l'option 4 : 5.294117647058823%
Pour l'option 5 : 6.470588235294119%
```

Note : Si l'on fait la somme des probabilités, on n'obtient pas 100 mais 99. Cela souligne la nécessité de bien paramétrer les contraintes et en particulier les multiplicateurs de Lagrange.

Ainsi, si quelqu'un souhaitait améliorer notre programme, c'est cette paramétrisation qu'il faudrait modifier.

#### 4. Comparaison des résultats

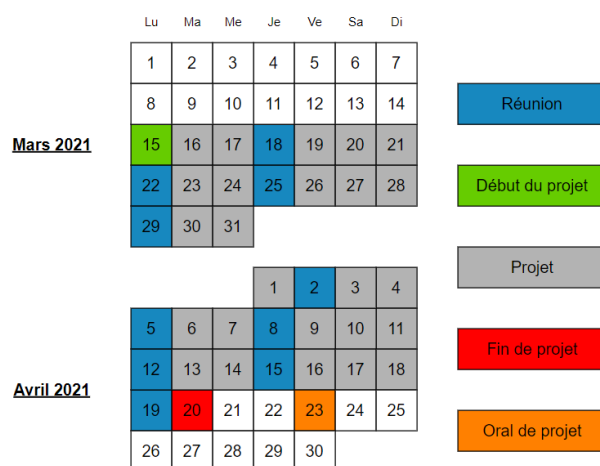
Après avoir étudié les résultats entre les deux programmes, On remarque que la gestion des contraintes est mieux réalisée et est plus simple à mettre en place avec le programme classique qu'avec celui quantique. Cela est notamment dû à la complexité du paramétrage des multiplicateurs de Lagrange utilisés pour le programme quantique tandis que le programme classique n'a pas cette difficulté.

On peut qualifier également le programme classique de plus simple d'utilisation car le programme quantique nécessite d'utiliser l'IDE Leap. Toutefois le programme quantique permet de proposer plusieurs solutions car celles-ci sont différentes à chaque compilation, chose que n'est pas du tout le cas pour le programme classique qui lui offre un panel de solutions plus limité.

Enfin le programme classique est plus rapide que le programme quantique mais cela est à relativiser car pour un nombre d'étudiants ou de projets supérieur, le programme classique pourrait montrer ses limites comme vu avec la courbe trouvée en travaillant sur max-cut.

## IV. Gestion de projet

Tout d'abord pour notre projet, nous avons convenu de faire deux réunions par semaine, les lundis et jeudis à 14h30. Les réunions duraient entre 30 minutes et une heure et nous permettaient de discuter avec notre responsable projet, lui faire part de nos avancées ainsi que de nos problèmes.



Après chaque réunion, nous faisons une « to-do list » récapitulant les tâches à faire permettant ensuite de les répartir entre nous. Cela nous a permis d'avoir une bonne vision à chaque étape et une vision globale du projet car on pouvait répondre aux 3 questions importantes :

- Qu'est-ce que l'on a fait ?
- Qu'est-ce que l'on fait actuellement ?
- Qu'est-ce que l'on doit faire ?

Et en regroupant les « to-do list » ensemble (Annexe n° 5), nous avons pu par la suite réaliser un diagramme de Pertt (Annexe 6) et un diagramme de Gantt (Annexe 7) permettant de visualiser encore plus concrètement notre projet.

Donc la gestion de projet a été un élément important de notre projet car cela nous a permis de répartir efficacement les tâches à réaliser tout en respectant les deadlines.

## Conclusion

Finalement, nous nous sommes rendu compte que la question à se poser n'était pas « Quel ordinateur est meilleur ou plus rapide que l'autre » mais plutôt « Quelle est la taille du problème à traiter ? » car d'après nos différents résultats, les plus petits problèmes seront traités plus rapidement et trouveront la meilleure solution avec un ordinateur classique, mais imaginons le cas d'une répartition de projet avec 10 000 étudiants, alors l'ordinateur quantique sera à privilégier car plus intéressant.

De plus, nous avons réussi tous les objectifs que nous nous étions fixés au départ, c'est-à-dire traiter les deux problèmes du projet: celui Max-cut et celui de répartition des projets et d'ensuite comparer leurs résultats entre l'ordinateur classique et l'ordinateur quantique pour en tirer des conclusions.

Grâce aux échanges par mail avec D-Wave et l'intérêt qu'ils portent à notre problème de répartition, notre projet peut potentiellement ouvrir la voie à un partenariat entre l'ISEN et D-Wave au travers d'abonnements gratuits par exemple pour une utilisation plus longue et approfondie de leurs ordinateurs quantiques et ainsi permettre un développement de ce domaine en recherche et en développement informatique dans l'école.

Également, cela fut très intéressant car la deuxième partie du projet concernant la répartition des projets avait un besoin concret qui était de faciliter la répartition des projets à notre cliente Pascale DIENER mais également aux futurs gérants de la répartition des projets M1 donc il y avait non seulement un aspect réaliste mais surtout une vraie utilité sur le long terme.

---

## Bibliographie

Site de D-Wave :

<https://www.dwavesys.com/>

Accès pour Leap :

<https://cloud.dwavesys.com/leap/>

Site de D-Wave sur l'architecture d'un QPU :

[https://docs.dwavesys.com/docs/latest/c\\_gs\\_4.html](https://docs.dwavesys.com/docs/latest/c_gs_4.html)



## Table des annexes

<u>Annexe 1 : Echanges avec D-Wave.....</u>	<u>p.30</u>
<u><i>Annexe 2 : Notice du programme version classique.....</i></u>	<u><i>p.34</i></u>
<u>Annexe 3 : Notice du programme version quantique.....</u>	<u>p.36</u>
<u>Annexe 4 : Documentation pour programme classique.....</u>	<u>p.38</u>
<u>Annexe 5 : Documentation du programme quantique.....</u>	<u>p.42</u>
<u>Annexe 6 : Liste des tâches.....</u>	<u>p.45</u>
<u>Annexe 7 : Diagramme de Pertt.....</u>	<u>p.46</u>
<u>Annexe 8 : Diagramme de Gantt.....</u>	<u>p.47</u>

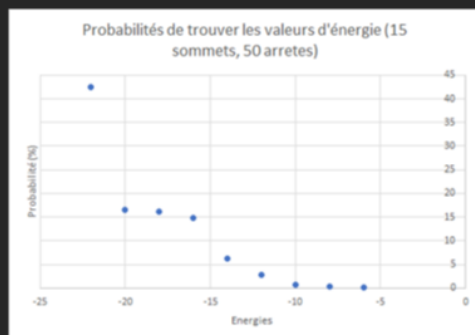
## Annexe 1 : Echanges avec D-Wave

To whom it may concern,

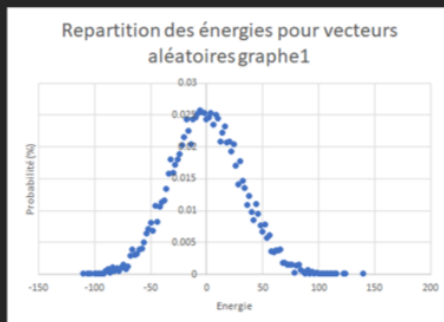
I hope this message finds you well. My name is Charles Deleforge, I am a French engineering student from ISEN and I am contacting you on behalf of my work group as we are currently working on a project using leap's IDE. We have a number of questions and we would be glad if you could bring an answer to them.

First off about our project, we worked first on the Max Cut problem and are now transitioning to a group repartition problem. We thought the later could interest you, as there is nothing like it currently in your examples list.

Concerning our questions, the first one is that we were wondering if the solver is seeking every possible solution? More precisely, how does it work for the lowest energy solution? Does it just ignore the path towards any solution once it has been found? (the graph beneath shows the probability of finding various energy values)



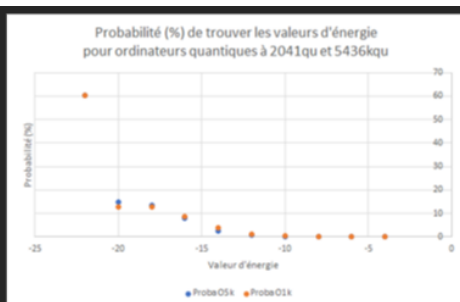
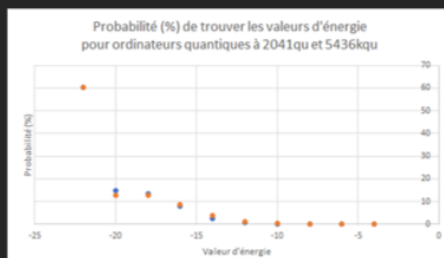
We also wanted to have a clarification about the how the quantic computer works. When using a quantic solver and plotting the results we obtain a gauss like graph. Shouldn't we instead have a decreasing curve, the lower energy solution being the most probable one?



The previous graph is the results we were expecting when simulating with a classical computer. As you can see, results are going from around -140 to 140. (ajouter avant-dernier graphe de #idée) However, once we run our program in a quantic solver, the range of result is far from what we simulated. Which is why we are wondering if an offset is added, or if the result of the objective function is calculated in a different way? Our function is the following:

$$\sum_{i,j}^n Z_i Z_j \text{ if there is an edge between } i \text{ and } j$$

where  $\begin{cases} Z_i = 1 \text{ if summit } i \in A \\ Z_i = -1 \text{ if summit } i \in B \end{cases}$



An other question about a graph, we just wanted to know what was the difference between the two QPU available on Leap (advantage\_system1.1 and DW\_200Q\_6). Are their purpose totally different or are they two different ways of approaching a submitted problem?

And finally, we wanted to ask if you knew of a method that could allow us to find the minimal number of anneals needed in order to find the optimal solution?

Thank you so much for your attention and answers

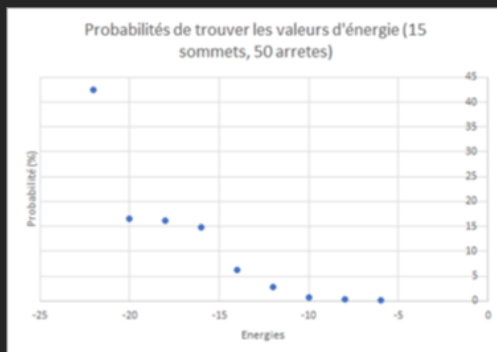
Best regards,  
Charles DELEFORGE

Hi Charles,

Thank you for clarifying. I am really glad to know you are having a positive experience using our system. About the problem you are trying to solve (group repartition), is it a variation of graph partition? If so, you might find the [graph partition](#) example in our collection of resources helpful.

Here are my thoughts for your question:

Concerning our questions, the first one is that we were wondering if the solver is seeking every possible solution? More precisely, how does it work for the lowest energy solution? Does it just ignore the path towards any solution once it has been found? (the graph beneath shows the probability of finding various energy values)



Our QPU uses quantum annealing to solve a problem which relies on the fundamental rule of physics that everything tries to find its minimum energy state.

When the anneal begins, all the qubits are in superposition and have an equal probability of being 0 or 1. During the anneal, this probability is controlled by external magnetic field which is defined by the linear and quadratic biases of your problem. So, the energy landscape changes over time as seen here:

[https://docs.dwavesys.com/docs/latest/c\\_gs\\_2.html#change](https://docs.dwavesys.com/docs/latest/c_gs_2.html#change). By the end of the anneal, each qubit takes either 0 or 1 while trying to minimize the energy.

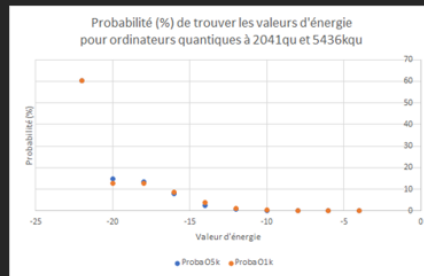
The affect is that the QPU doesn't follow a path to evaluate all solutions to the problem. Instead it seeks to stay in the lowest energy state (which corresponds to the optimal solution to your problem) as the anneal happens, or as the energy landscape changes around the lowest point. You can read more about it [here](#).

Also, you might this [video](#) helpful.

I also want to point that due to the probabilistic nature of the QPU you do not always get the minimum energy solution.



Again for this question, it would help to have more information about the setup. Would you mind sharing how you got a range of -140 to 140 as the possible solution space? Did you get a different energy from the quantum computer than manually solving the problem with the returned values for a sample?



An other question about a graph, we just wanted to know what was the difference between the two QPU available on Leap (advantage\_system1.1 and DW\_2000Q\_6). Are their purpose totally different or are they two different ways of approaching a submitted problem?

Both Advantage\_system1.1 and DW\_2000Q\_6 are different generation of our QPU. They both solve the problem using quantum annealing. Some of the main difference between the two chips include but is not limited to: number of qubits (Advantage\_system1.1: 5000+, DW\_2000Q\_6: 2000+), connectivity between qubits (Advantage\_system1.1: each qubit is connected to upto 15 other qubits, DW\_2000Q\_6: each qubit is connected to upto 6 other qubits) and [topology](#) (Advantage\_system1.1: Pegasus, DW\_2000Q\_6: Chimera).

And finally, we wanted to ask if you knew of a method that could allow us to find the minimal number of anneals needed in order to find the optimal solution?

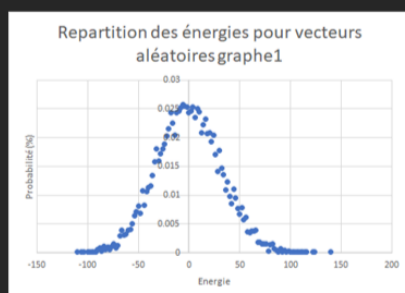
Unfortunately, there is no fixed method to find the optimal value for num\_reads. Though higher num\_reads increase your probability of success, it also costs you more in terms of qpu time. The optimal value is very problem specific and it needs some exploring. For example, if you are working on a smaller problem with limited sample space you might be able to get good results with smaller num\_reads where as for harder problems you might need to use a higher value.

Best Regards,

Tanvi Mittal  
D-Wave Systems Inc.  
[www.dwavesys.com](http://www.dwavesys.com)

We also wanted to have a clarification about the how the quantic computer works. When using a quantic solver and plotting the results we obtain a gauss like graph. Shouldn't we instead have a decreasing curve, the lower energy solution being the most probable one?

It is difficult to answer this question without knowing more about the results that you are seeing from the quantum computer and how you are calculating the probability. Would you be able to share the [histogram from the problem inspector](#)? Here is how you can use the inspector to capture this information: [https://docs.ocean.dwavesys.com/en/stable/examples/inspector\\_graph\\_partitioning.html](https://docs.ocean.dwavesys.com/en/stable/examples/inspector_graph_partitioning.html).



The previous graph is the results we were expecting when simulating with a classical computer. As you can see, results are going from around -140 to 140. (ajouter avant-dernier graphe de #idée) However, once we run our program in a quantic solver, the range of result is far from what we simulated. Which is why we are wondering if an offset is added, or if the result of the objective function is calculated in a different way? Our function is the following:

$$\sum_{i,j}^n Z_i Z_j \text{ if there is an edge between } i \text{ and } j$$

$$\text{where } \begin{cases} Z_i = 1 \text{ if summit } i \in A \\ Z_i = -1 \text{ if summit } i \in B \end{cases}$$

Again for this question, it would help to have more information about the setup. Would you mind sharing how you got a range of -140 to 140 as the possible solution space? Did you get a different energy from the quantum computer than manually solving the problem with the returned values for a sample?

## Annexe 2 : Notice du programme version classique

### Comment utiliser le programme de répartition de projets

#### Étape n°1 :

Installez GLPK en utilisant le guide que vous pourrez trouver joint dans le dossier.

#### Étape n°2 :

Assurez vous que l'extension du document Excel sur lequel se trouvent les réponses des étudiants soit un ".XLS"

#### Étape n°3 :

Dans le code du programme, ligne 58 du script solveProjectDistribution.py, entrez le chemin menant à la localisation de votre fichier Excel

#### Étape n°4 :

Vous pouvez aussi remplir les places par projet directement sur le programme, ligne 87 du script solveProjectDistribution.py en remplissant la variable nbStudentPerProject. Vous pouvez également remplir ces valeurs plus tard dans le programme.

#### Étape n°5 :

Vous pouvez maintenant exécuter le programme en entrant la commande "python solveProjectDistribution.py" dans votre terminal

#### Étape n°6 :

Entrez les informations liées au nombre d'étudiants et projets.

#### Étape n°7 :

Le programme entame une vérification ; les étudiants n'ayant pas rempli correctement le formulaire sont affichés sur votre écran

#### Étape n°8 :

Si vous avez choisi de ne pas faire l'étape 4, vous pouvez choisir de remplir le nombre d'étudiants max autorisé pour chaque projet. Si vous avez rempli la variable nbStudentPerProject, indiquez n

#### Étape n°9 :

Procédez ensuite à une vérification de vos données puis si celles-ci sont correctes, appuyez sur y, sinon appuyez sur n

#### Étape n°10 :

Le modèle est prêt, il est d'abord transformé en fichier Linear Programming. Celui-ci est exporté puis envoyé au solveur GLPK pour être résolu. Sur votre terminal s'affiche les informations retournées par GLPK ainsi que le temps d'exécution

#### Étape n°11 :

Plusieurs fichiers de résultats vont être bientôt exportés, choisissez un nom qui servira de racine à ces fichiers

#### Étape n°12 :

Par la suite, plusieurs choix s'offrent à vous :

- En appuyant sur A, le nombre d'étudiants pour chaque projet s'affiche en console et un fichier du nom entré et la terminaison "nbStudent.txt" est exporté.
- En appuyant sur Z, la répartition des étudiants pour chaque projet s'affiche en console et un fichier du nom entré et la terminaison "all.txt" est exporté.
- En appuyant sur E, vous pouvez rechercher la répartition des étudiants pour un projet en particulier. Celui-ci doit être indiqué par son numéro.
- En appuyant sur T, vous exporterez un fichier permettant d'utiliser le programme quantique. Il contient les options des étudiants, leurs identifiants et le nombre de place par projet. Son nom est le nom entré avec la terminaison "toQuantumProgram.txt"
- En appuyant sur Q, vous quitterez le programme.

#### Mot du développeur :

Pour toute question sur l'utilisation de ce programme, vous pouvez joindre notre groupe.

## Annexe 3 : Notice du programme version quantique

### Comment utiliser le programme de répartition de projets

Attention ce programme nécessite d'avoir déjà utilisé sa version classique et exporté le fichier finissant par "toQuantumProgram.txt"

Note : A chaque compilation, vous obtiendrez des répartitions différentes, c'est l'intérêt des ordinateurs quantique à annealing !

#### Étape n°1 :

Créez un compte Leap sur le site : <https://cloud.dwavesys.com/leap/signup/>.

Celui-ci sera valide pendant 1 mois. Vous avez le droit à 1 minute sur les ordinateurs quantiques et 20 minutes sur les ordinateurs hybrides.

#### Étape n°2 :

Après avoir vérifié votre adresse mail et vous être connecté, dirigez vous vers l'IDE Workspaces. Chargez un des exemples puis extrayez l'ensemble des fichiers de l'archive RepartitionProjetQuantique.rar en les glissant dans l'IDE (Integrated Development Environment).

#### Étape n°3 :

Copiez l'intégralité du contenu du fichier finissant par "toQuantumProgram.txt" (exporté préalablement grâce au programme classique) dans le fichier "dataFromClassicProgram.py".

#### Étape n°4 :

Vous pouvez maintenant exécuter le programme en vous plaçant dans le fichier main.py et en cliquant sur le bouton run situé en haut à droite de l'IDE.

#### Étape n°5 :

Entrez les informations liées au nombre d'étudiants et projets.

#### Étape n°6 :

Ensuite, choisissez l'ordinateur sur lequel s'effectue les calculs. Si vous choisissez l'ordinateur quantique, appuyez sur Q sinon appuyez sur une autre touche.

Note : Pour des problèmes de taille conséquente (>100 étudiants) nous vous conseillons d'utiliser l'ordinateur hybride. En effet, l'ordinateur quantique peut avoir des difficultés au moment de l'intégration.

#### Étape n°7 :

Vous pouvez également choisir de sauvegarder les données si vous souhaitez garder un trace des temps d'exécution, de la matrice Q et des résultats. Si c'est le cas, appuyez sur Y, sinon appuyez sur une autre touche.

Note : Le fichier "save" sera créé, son nom est suivi d'un identifiant unique afin que vous puissiez exporter à nouveau sans l'écraser

#### Étape n°8 :

Par la suite, plusieurs choix s'offrent à vous, comme pour le programme classique :

- En appuyant sur A, le nombre d'étudiants pour chaque projet s'affiche en console et un fichier commençant par "results" et terminant par "nbStudent.txt" est exporté.
- En appuyant sur Z, la répartition des étudiants pour chaque projet s'affiche en console et un fichier commençant par "results" et terminant par "all.txt" est exporté.
- En appuyant sur E, vous pouvez rechercher la répartition des étudiants pour un projet en particulier. Celui-ci doit être indiqué par son numéro.
- En appuyant sur Q, vous quittez le programme.

Note : Les fichiers sont là aussi créés avec un identifiant unique (lié à l'heure) pour ne pas écraser les données.

Mot du développeur :

Pour toute question sur l'utilisation de ce programme, vous pouvez joindre notre groupe.

## Annexe 4 : Documentation pour programme classique

### Documentation Project Distribution Classic Program

---

- solveProjectsDistribution.py

Main script to launch the distribution of students

```
def getOptionsFromXLS(students, nbProjects):  
  
    Get choices of students from the choices made in the XLS file  
  
    :param students: list of students  
    :param nbProjects: number of projects  
  
    :type students: list of integers  
    :type nbProjects: integer  
  
    :return w: options of the students  
    :rtype w: dict {(student, project) : weight of options}
```

- fromExcelToPython.py

Script allowing to get the information of the students choices from a XLS file

```
class Student:  
    Student class corresponding to the student in M1 at ISEN  
  
    :member id: identifiant of the student  
    :member option1: first choice of the student  
    :member option2: second choice of the student  
    :member option3: third choice of the student  
    :member option4: fourth choice of the student  
    :member option5: fifth choice of the student  
  
def getStudentsFromXls(fileXls, numberProjects, numberStudents)  
  
    Get the list of student and their options from a xls file  
  
    :param fileXls: the name of the file to get the option of each students  
    :param numberProjects: the number of projects  
    :param numberStudent: the number of students  
  
    :type fileXls: string  
    :type numberProjects: int  
    :type numberStudents: int  
  
    :return students: list of students with their choce of options  
    :return studentsID: list of the name of the students  
  
    :rtype students: table of Student  
    :rtype studentsID: table of string
```

- lp.py

Export a lp file from a model

```
class Student:

    Student class corresponding to the student in M1 at ISEN

    :member id: identifiant of the student
    :member option1: first choice of the student
    :member option2: second choice of the student
    :member option3: third choice of the student
    :member option4: fourth choice of the student
    :member option5: fifth choice of the student

class Project:

    Project class corresponding to the project proposed by ISEN

    :member id: identifiant of the project
    :member numberOfStudentAllowed: number of student allowed for this project

class Model:

    Model class constructed with a list of students and a list of projects

    :member students: list of students from class Student
    :member projects: list of projects from class Project

def printModel(model):

    Print the information of a model

    :parameter model: model corresponding to the current problem
    :type model: class Model

def lp(model :Model, fileName):

    Transform a model into a lp file

    :parameter model: model corresponding to the current problem
    :type model: class Model

    :parameter fileName: name of the lp file to be written
    :type fileName: string

    :return lpFile: lpFile corresponding to the current problem
    :rtype lpFile: file (.lp => Linear Program)
```

- findSolution.py

Script to launch the solver glpk in the consol

```
def findSolution(lpFile, solutionFile):  
  
    Launch the command in windows shell to find the solution of the lp file  
  
    :param lpFile:  
    :param solutionFile:  
  
    :type lpFile:  
    :type solutionFile:  
  
    :return solutionFile: the file of the solution
```



## - readSolution.py

Read and interface the solution found by the solver GLPK

```
class ProjectSolved:

    Project class corresponding to the project divided by student according to
    their choices

    :member id: identifiant of the project
    :member students: list of the students for this project
    :member room: room planned for this project

def readSolution(fileName, nbStudents, nbProjects, room):

    Interfaces the solutions from the result of lp file after using glpk

    :parameter fileName: Name of the file to interface
    :parameter nbStudents: number of students
    :parameter nbProjects: number of projects
    :parameter room: nb of student allowed per project

    :type fileName: String
    :type nbStudents: integer
    :type nbProjects: integer
    :type room: table of integer

def interface(projects :ProjectSolved, fileName, students: Student, studentID, w,
room):

    Interfaces the results of the division of projects by students

    :param projects: list of the projects
    :param fileName: name of the file to save the results
    :param students: list of the students with the options
    :param studentID: list of name of students
    :param w: options of the students
    :param room: number of students allowed per projects

    :type projects: class Porject
    :type fileName: String
    :type students: table of Student
    :type studentID: table of String
    :type w: dictionary of {(student, project) : integer}
    :type room: table of integer

    :return files: the display in files saved

def calculateProba(projects : ProjectSolved, students):

    Calculate the different distributions of students for each options

    :param projects: list of the projects
    :param students: list of the students with the options

    :type projects: class Project
    :type students: table of Student

    :return files: the display in files saved
```

## Documentation Project Distribution Quantum Program

---

- main.py

Main script to launch the distribution of students with the D-wave quantum computer

- display.py

```
def interface(projects :ProjectSolved, fileName, students: Student, studentID, w):

    Interfaces the results of the division of projects by students

    :param projects: list of the projects
    :param fileName: name of the file to save the results
    :param students: list of the students with the options
    :param studentID: list of name of students
    :param w: options of the students

    :type projects: class Porject
    :type fileName: String
    :type students: table of Student
    :type studentID: table of String
    :type w: dictionary of {(student, project) : integer}

    :return files: the display in files saved


def calculateProba(projects : ProjectSolved, students):

    Calculate the different distributions of students for each options

    :param projects: list of the projects
    :param students: list of the students with the options

    :type projects: class Project
    :type students: table of Student

    :return files: the display in files saved
```

- hybridProgUsefulFunctions.py

Useful functions to launch the program

```
def getListOfStudent(n_students):

    Get the list of student

    :param n_students: number of the students
    :type n_students: integer

    :return students: list of students
    :rtype students: list of integer

def roomWithCoeff(rooms, coeff):

    Add a coefficient to a list of room

    :param rooms: list of rooms
    :param coeff: coefficient to multiply each element of the list

    :type rooms: list of integer
    :type coeff: integer

    :return roomCoeffed: list of room with coefficient
    :rtype roomCoeffed: list of integer

def getIndex(student_index, project_index, n_projects):

    Get the index of the matrix corresponding to the index of students and
    projects
    (opposite of get_student_and_project)

    :param student_index: index of the student
    :param project_index: index of the project
    :param n_projects: number of projects

    :type student_index: integer
    :type project_index: integer
    :type n_projects: integer

    :return index: index corresponding to project_student
    :rtype students: integer

def getStudentAndProject(index, n_projects):

    Get the indexes of the matrix corresponding to the index of students and
    projects
    (opposite of get_index)

    :param index: index of the project_student
    :param n_projects: number of projects

    :type index: integer
    :type n_projects: integer

    :return (student_index, project_index): index corresponding to project_student
    :rtype (student_index, project_index): tuple of integers

def getProjectsSolvedFromSched(n_projects, tmpProjects, room):

    Get the projects from a list

    :param n_projects: number of projects
    :param tmpProjects: dictionary of project not sorted
    :param room: list of number of student allowed per project

    :type n_projects: integer
    :type tmpProjects: dictionary
    :type room: list of integer

    :return projects: dictionary of projects associated to students
    :rtype projects: dictionary of integers

def changeW(w):

    Change the weight of each options

    :param w: {(student, project) : weight}, initial weights
    :type w: {(integer, integer) : integer}

    :return w: param w with the good weights
    :rtype w: {(integer, integer) : integer}
```

## - Student.py

```
class Student:
    Student class corresponding to the student in M1 at ISEN

    :member id: identifiant of the student
    :member option1: first choice of the student
    :member option2: second choice of the student
    :member option3: third choice of the student
    :member option4: fourth choice of the student
    :member option5: fifth choice of the student
```

## - ProjectSolved.py

Export a lp file from a model

```
class ProjectSolved:

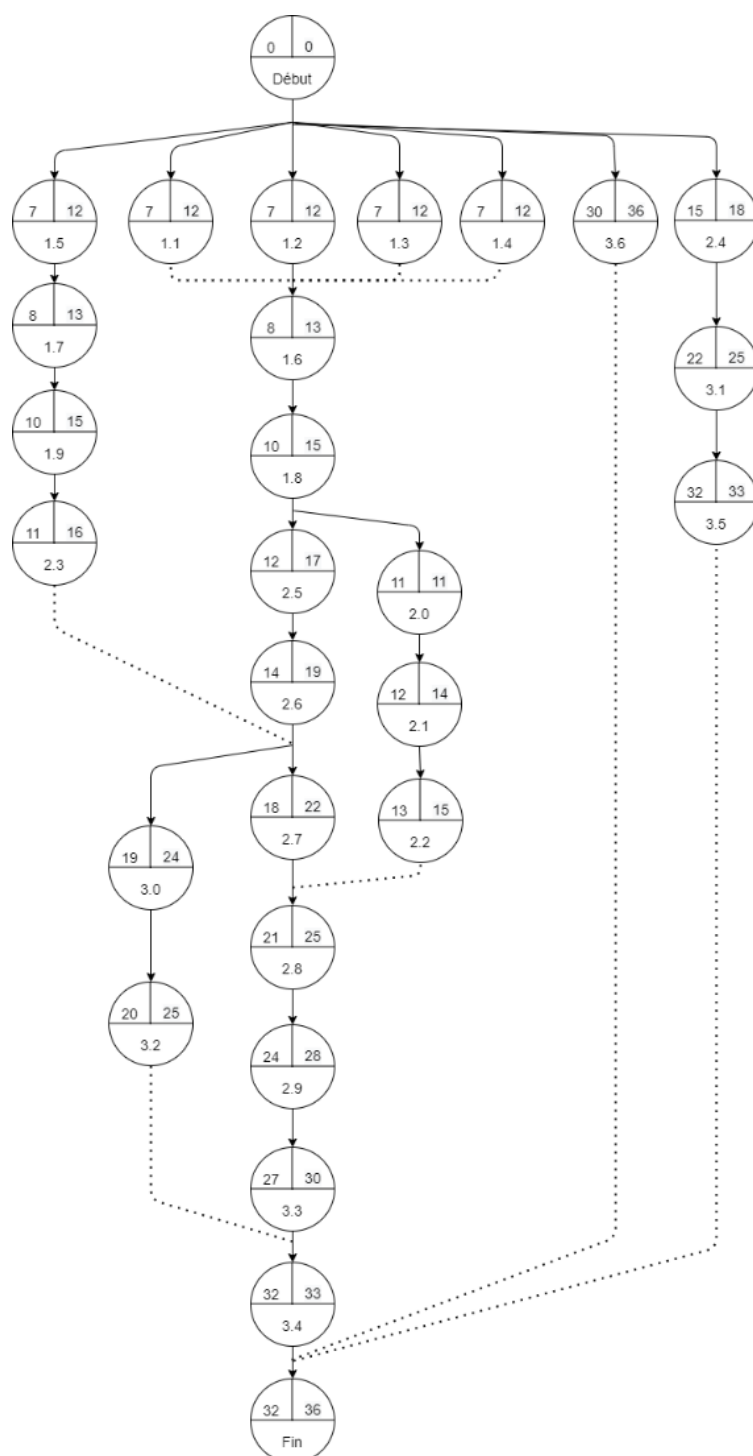
    Project class corresponding to the project divided by student according to
    their choices

    :member id: identifiant of the project
    :member students: list of the students for this project
    :member room: room planned for this project
```

## Annexe 6 : Liste des tâches

- 1.1- Trouver les limites (sommet, arête...) de chaque calculateur (même BinaryQuadraticModel)
- 1.2- Calculer les différents temps d'exécution et donner une approximation mathématique (temps exponentiel...)
- 1.3- Calculer les probas (pour la meilleure valeur d'énergie et les deux premières), espérance, et espérance de lancer pour être sûr à 98-99% d'avoir une (très) bonne solution.
- 1.4- Réfléchir à la conception du graphe de la répartition des projets.
- 1.5- Regarder le code plus en détail, voir le code du constructeur de BinaryQuadraticModel, ExactSolution... => voir si c'est réimplémentable (et si la limite des sommets vient de là) + librairies existantes.
- 1.6- Regarder les deux vidéos de D-Wave comme introduction aux modèles mathématiques.
- 1.7- Temps d'ordinateurs hybrides + Etudier le fonctionnement ordis hybrides.
- 1.8- Modélisation mathématiques du problème de distribution des projets.
- 1.9- Voir si le nombre de qubit influe sur le temps ou la qualité des solutions.
- 2.0- Demander à Diener un cahier des charges.
- 2.1- Faire un cahier des charges.
- 2.2 - Envoyer le cahier des charges à Mme Diener.
- 2.3 - Rédiger un premier brouillon de mail à Dwave.
- 2.4 – Etape 1 – Démarrage du rapport de conduite de projet.
- 2.5- Créer le fichier lp.
- 2.6 - Implémenter un algo de vérification.
- 2.7 - Récupérer la solution des .lp sur windows avec glpk .
- 2.8 - Créer un script python permettant de transcrire les données d'entrée de Diener en les classes utilisé par le générateur de .lp.
- 2.9 - Créer un script python pour interfacer les résultats sortis par glpk (ou autre) => nécessite l'étape 2.7.
- 3.0 - Commencer les recherches sur la méthode de résolution quantique, utilisation des ordis hybrides, calculs du multiplicateur de Lagrange.
- 3.1 – Etape 2 – Planification du rapport de conduite de de projet
- 3.2 - Faire valider le mail pour D-wave et l'envoyer
- 3.3. - Trouver d'autres solutions avec glpk.
- 3.4 - Voir si on a la meilleure solution (=> comparer avec l'ordi classique pour le max de sommet possible)
- 3.5 – Etape 3 – Exécution du rapport de conduite de projet.
- 3.6 – Rapport global du projet.

## Annexe 7 : Diagramme de Pertt



## Annexe 8 : Diagramme de Gantt

