Quantum Annealing with D-Wave Machines

Quantum Computing (M1)

Practical lab

(max: 4 people per group)

Sloths[1] moon colonization

pic: © L.T.GraphicArtist

Real-life context. Fifty years had passed since the last animal went to the moon[2]. Lately, the sloths have shown their interest to colonize it[3]. Indeed, they stop being lazy today in order to be more lazy tomorrow. Their goal: to be weightless for the rest of their life.

[1] *fr: le paresseux (l'animal).*
[2] Apollo 17, taking place between 7 and 19 December 1972.
[3] For those who can be surprised by this news: so far, they were too lazy to show us they have the same intelligence.

JUNIA ISEN

Samuel Deleplanque. (prénom.nom@junia.com)

OVERVIEW

The infrastructure on the moon had been built. There are **2** different places for working and **3** separate dormitories for a total of **6** sloths. The remaining work[4] for this operation (i.e., **Part1** and **Part2**) is about splitting the sloths group into subgroups:

- **Part1**: During daytime, the group of 6 sloths must be split into 2 different groups for working: one for each of the 2 different locations. The separation will be made according to the first criterion **c1** defined below.
- **Part2**: During the night, the 6 sloths must be placed in one of 3 dormitories according to a second criterion **c2** and to a number of beds available in each dormitory.

The two criteria can be defined and placed in their context:

- **c1: laziness risk.** Among the set of sloths, some pairs of them have a high risk of doing nothing if they work together. Thus, the separation of such pairs into two different groups (i.e., to work in different places) must be done as much as possible. For a given pair of sloths, the values related to the *laziness risk* can be defined as follow:
    - the more $c1_{ij}$ is high the more there is a risk of laziness if the sloths i and j work together.
- **c2: genetic heterogeneity.** The sloths will be also separated to create 3 groups: one per dormitory. These subgroups must be made in order to have genetic heterogeneity. For a given pair of sloths, the values related to the *heterogeneity* can be defined as follow:
    - the smaller the $c2_{ij}$ the more there is heterogeneïty if the sloths i and j sleep in the same dormitory.

The data related to the population and to the criteria **c1** & **c2** is given in the following section.

[4] actually: your work.

DATA

Because we are limited in time, we will work with a small population of **6** sloths. The data for Part1 (c1) is given in Figure 1. For Part2, Figures 2 and 3 give all the information: the values related to the criterion c2 and the number of beds per dormitory.

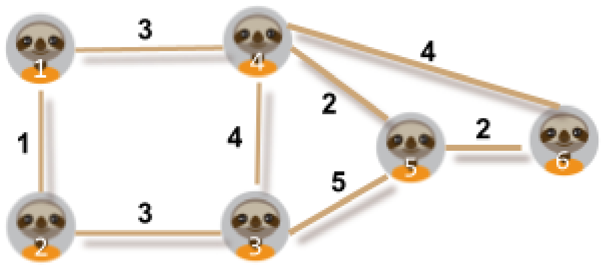


Figure 1. Laziness risk for Part1. Here, the valuation of each edge, if it exists, is the laziness risk: <u>the bigger it is, the more the risk is high</u>. For example, the sloths (3) and (4) have the highest risk of laziness.

i \ j	1	2	3	4	5	6
1	0	1	6	1	8	0
2		0	3	3	9	8
3			0	8	7	4
4				0	9	8
5					0	9
6						0

Figure 2. Genetic heterogeneity for Part2. Here: <u>the lesser it is, the bigger the heterogeneity will be</u>. This criterion is symmetric: $c2_{16} = c2_{61} = 0$ means the sloth 6 and the sloth 1 have the largest heterogeneity values while $c2_{24} = c2_{42} = 9$ means sloth 2 and 4 have the smallest genetic heterogeneity.

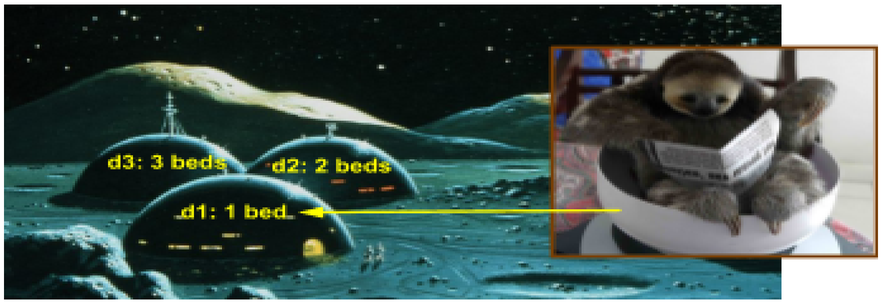


Figure 3. Capacities of the 3 dormitories d_1, d_2 and d_3

Part1

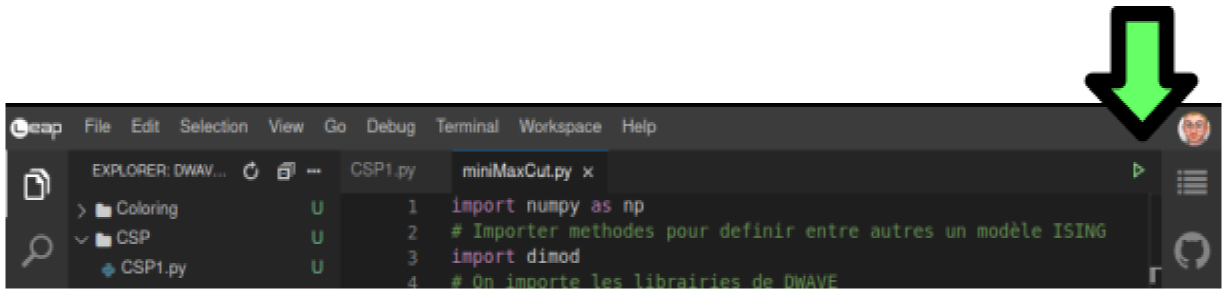
i. What is the optimization problem related to part 1[5]?

ii. Formulate the Objective function from the graph given in Figure 1 by considering that the edges of the graph have no valuation (it is like to consider all integers on the edges equal to 1).

iii. Imagine one solution of the problem without trying to find the best one, and report it after calculating the Objective function you obtain for your solution. To report the solution, give the two sets of sloths.

Optimizing with D-Wave (Part1).

iv. Launch the execution of the given file miniMaxCut.py[6] by clicking on the green arrow (see Figure 4). And report the result you obtain (screenshots).



```
import numpy as np
# Importer methodes pour definir entre autres un modèle ISING
import dimod
# On importe les librairies de DWAVE
```

Figure 4. Green arrow to launch the calculation through Leap, the online tool (usable with your browser).

[5] You have the slides on Teams...And not a lot of optimization problems to check…And you can also read the rest of this subject to be sure.

[6] The name of the file might be an important clue about one of the previous questions…

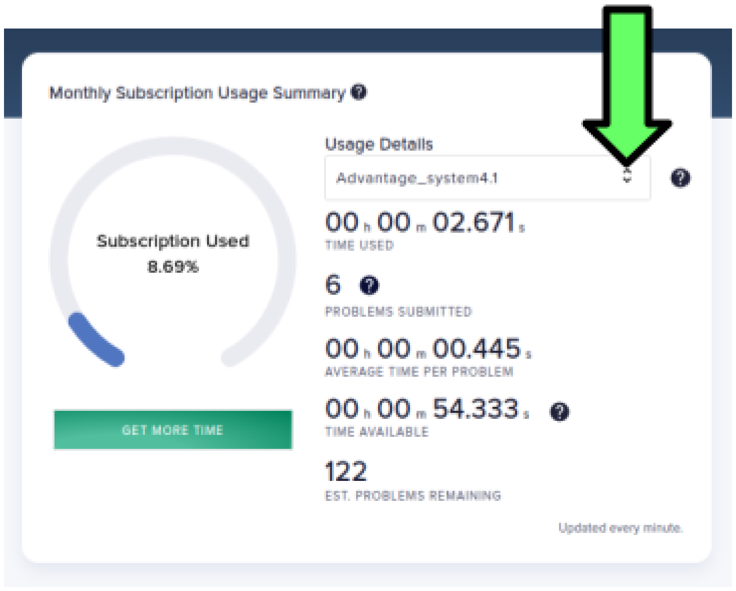
v. Using the portions of code provided in miniMaxCut.py[7], implement your python code for the problem of Part1 without valuation. You will find lots of help in the last slides of the Quantum Annealing course. Run your code on a D-Wave machine and report the results[8]. Also note the best solution obtained and find the detail of the calculation which gives the value of the Objective function (the energy column).

vi. Carry out the same work but this time with the values of the arcs. This time: try to calculate the Max-Cut between the sets of clothes[9].

vii. Modify your code in order to solve with a more recent quantum computer by integrating the following instruction at the right place[10] :

```
sampler = EmbeddingComposite(DWaveSampler(solver='Advantage_system4.1'))
```

To go further: In https://cloud.dwavesys.com/leap/, you have access to your Dashboard with some data related to the machines you can use. It seems the Advantage 5.1 exists but is not available at the time this subject is written. Feel free to try other machines.


Monthly Subscription Usage Summary

Usage Details

Advantage_system4.1

00 h 00 m 02.671 s
TIME USED

6
PROBLEMS SUBMITTED

00 h 00 m 00.445 s
AVERAGE TIME PER PROBLEM

00 h 00 m 54.333 s
TIME AVAILABLE

122
EST. PROBLEMS REMAINING

Subscription Used
8.69%

Figure 5. Machines available for calculation.

Report the result you obtain and check the differences or similarities with your previous result. You do not have to use the 4.1 machine specifically.

[7] The case initially in the miniMaxCut.py file is related to a triangle shape graph where the edges are (0,1), (1,2), (2,0) without valuation (':1'). Do not forget : here (i.e., for the model we used in Part1), **the binaries are {-1;+1} values**.

[8] With a screenshot of the resulting terminal.

[9] You will see some examples in the course: some slides represent such a cut with scissors and a dotted line. You just have to sum the values on the edges of the cut.

[10] In https://cloud.dwavesys.com/leap/, you have access to your Dashboard with some data related to the machines you can use. It seems the Advantage 5.1 exists but is not available at the time this subject is written. Feel free to try other versions.

Part2

Mathematical formulation of the problem. Maximizing the genetic heterogeneity of the 3 groups sleeping in different dormitories consists in creating 3 subgroups minimizing the total sum of all $c2_{ij}$ related to all the pairs (i,j) for slots in each group. To express that mathematically, we note d the index of the dormitory ($d = 1..3$), i and j the index of two sloths, and the constant 6 the total number of sloths. The x_{id} correspond to binary variables[11] such that :

- $x_{id} = 1$ if the sloth i is sleeping in the dormitory d, $x_{id} = 0$ otherwise.

The Objective function of the Part2, which corresponds of maximizing the genetic heterogeneity of in the 3 dormitories, can be modeled by the minimization of the following function f[12].

$$Objective:\ Minimize\ (f) = \sum_{i=1}^{5} \sum_{j=i+1}^{6} \sum_{d=1}^{3} c2_{ij} x_{id} x_{jd}$$

The goal is to find which variables $x_{id} = 1$ in order the minimize f. However, it is not enough. Instead of the problem treated in Part1, this one requires constraints. Indeed, a feasible solution needs to satisfy two series of them, where we note $Beds(d)$ the number of available beds in the dormitory d:

1. Bed number limit in each dormitory. It can be modeled as follow with 3 constraints:
    - $\sum_{i=1}^{6} x_{id} \leq Beds(d), \qquad \forall\, d = 1..3$
2. Each sloth must be linked to exactly one dormitory. It can be modeled as follow with 6 constraints:
    - $\sum_{d=1}^{3} x_{id} = 1, \qquad \forall\, i = 1..6$

[11] So what we are looking for in this Part2.
[12] If you have trouble understanding that j starts at i+1, try to link that with the fact the table given in Figure 2 is symmetric. Why would we do the double calculation for nothing?

viii. Report the Objective function and all the constraints you obtain for the case related to the data given above (Figure 2 and 3).

ix. Without trying to find the best solution, imagine one solution of the problem according to the model (Objective function and constraints) and the data given above. Report the variables values of all the x_{id}, the resulting Objective function, and the impact of the solution in all the constraints[13].

Optimizing with D-Wave (Part2).

Lately[14], it is possible to declare a quadratic model with constraints directly in the python code[15]. For the problem treated in this Part2 the way to declare our problem is quite different from the problem of Part1. Here we will not care about the Hamiltonien (J and h in miniMaxCut.py) since it is possible to directly formulate the Objective function and the constraints in the code.

x. Create a new file from scratch directly in the online Leap IDE. Import the following packages[16] and the declaration of our model object[17]:

```python
from dimod import ConstrainedQuadraticModel, CQM,  SampleSet

from dimod import Binary, quicksum

from dwave.system import LeapHybridCQMSampler

import numpy as np
```

Note that the type of the variables is furnished by the package `Binary` and the different sums of the Objective function and the constraints will be written with the `quicksum` function.

[13] e.g., for the first dormitory constraint you could report: 0 + 0 + 1 + 0 + 0 + 0 <= 1 or, e.g., for the first constraints of the second series: 0 + 1 + 0 = 1.
[14] end of 2021.
[15] Before that, all the constraints must have been relaxed. In short: in a relaxation process, all the constraints are integrated in the Objective function in a way that, if they are not satisfied, the Objective function will tend to take a value in the contrary direction of the optimization (penalties). The difficulty is to weigh each constraint in the Objective function. D-Wave proposes to do it, but it does not mean they succeed all the time.
[16] You may notice that you can use numpy!
[17] Advice: rewrite the code or at least redo the tabulation

xi. Integrate the data.

Use the same data structures as the ones given below. You just have to adapt them to our case (e.g., you have 6 sloths instead of 4).

```
totalNumberOfSloths = 4

numberOfDormitories = 2

c2 = [[0, 3, 4, 10], [3, 0, 2, 2], [4, 2, 0, 3], [10, 2, 3, 0]]

beds = [2, 2]
```

xii. Declare the variables.

We have seen that x_{id} is the binary variable for the sloth i and the dormitory d. To declare the 6*3 matrix for all the variables, you can use the following code:

```
x = {

   (i, d): Binary('x{}_{}'.format(i, d))

      for i in range(totalNumberOfSloths)

      for d in range(numberOfDormitories)}
```

xiii. Declare and furnish the Objective function.

The following code is a first way to declare the Objective function:

```
objective = quicksum(c2[i][j] * x[(i,d)] * x[(j,d)]

      for i in range(totalNumberOfSloths)

      for j in range(totalNumberOfSloths)

      for d in range(numberOfDormitories)   )

cqm.set_objective(objective)
```

Try to improve it according to the following remark:

- It is possible to reduce the number of sums. To have some ideas, compare it with the Objective function of the mathematical model.

Keep the same last instruction to set the Objective function to the problem (i.e., to `cqm`).

xiv. Declare the bed number limit constraints.

To add a constraint to our qcm object, there is the `add_constraint` function:

```
for d in range(numberOfDormitories):

   cqm.add_constraint(quicksum(x[(i,d)]

   for i in range(totalNumberOfSloths)) <= beds[d])
```

xv. Declare the dormitory affectation constraints.

```
for i in range(totalNumberOfSloths):

      cqm.add_constraint(quicksum(x[(i,d)]

      for d in range(numberOfDormitories)) == 1)
```

xv.Launch the calculation

In the following code, the call of the constructor `LeapHybridCQMSampler` gives the information that the computer used is hybrid. Instead of the machine used for Part1 which was a full quantum machine, here, the work is made between a quantum and a classical machine. Click on the documentation link for more information on `sample_cqm`.

```
cqm_sampler = LeapHybridCQMSampler()

sampleset = cqm_sampler.sample_cqm(cqm)
```

xvi. Display the results

To display your results, use the following instruction:

```
print(sampleset)
```

To go further: try to find the right functions in the documentation in order to display the full model (Objective & Constraints), and all the solution, especially here:

- https://docs.ocean.dwavesys.com/en/stable/docs_dimod/reference/sampleset.html#dimod.SampleSet. For instance if you are having difficulty showing all the results, `aggregate` can save you (but not only).
- https://docs.ocean.dwavesys.com/en/stable/docs_inspector/intro.html

xvii. Analyse the results

You obtain the results for a certain number of solutions. You have to report:

- one screenshot of the result (even if it takes one page or more in your report),
- analyze the result and give one of the best and feasible[18] solutions obtained,
- draw simply the 3 subgroups of sloths in their dormitory,
- you might rapidly understand that some solutions cannot be taken into account. Take one and explain why.

IF THAT WAS NOT ENOUGH WORK (bonus)

If you have more time, you might want to use the potential of one of the D-Wave quantum machines. It is possible with the work you have already done in part2. You just need to generate a case on a wider scale (e.g., 50 or 100 sloths). Use randomness and loops to gain some time (otherwise the c1 and c2 criteria will be too long to generate).

Moreover, you might obtain some results without knowing how to evaluate them (which will take too much time if you have to develop another method, this time running on a classical computer). But you can focus on the feasibility of your solutions according to the size of your cases.

ABOUT the REPORT & the CODE to return:

- one report per group, but do not forget to write all the names on the front page,
- the number of pages is (here) not important, just try to answer each question,
- it can be written in french or english; screenshots are welcomed,
- the type of the file must be .pdf,
- 3 files not zipped: the report and two code files (resp. for Part1 and Part2),
- send the files to `youwillfindmynamessomewhereinthisdoc@junia.com`,
- the deadline will be given soon.

[18] i.e., respecting all the constraints.

CONCLUSION

- Of course we could not reach a problem size that makes the quantum annealing computer competitive: too much data would have had to be manipulated and too much time would have had to be spent doing it. This type of machines has today some limits: the number of qubits is not bad, but the connectivity between them is low[19]. For example for 20 sloths, the connectivity of 15 of the Advantage D-Wave machine will not be enough (some costly operations will use plenty of qubits to simulate a 19 connectivity).
- You might have realized that the two problems considered in this work have quadratic formulations (here: the products of binaries). It is for a good reason: the quantum machines based on quantum annealing can handle “easily” such expressions while the process implemented with this type of formulation for running on classical computers is way more complex.
- You worked on 2 optimization problems: a max-cut problem and a clustering problem. The sloths are not the only reasons to know how to solve them. For example, for the clustering problem (about the beds), it is the same problem when you want to assign packages to a fleet of delivery vehicles (the number of beds is the capacity of the vehicles). Moreover, if you already know how to solve the Traveling Salesman Problem[20], you have good tools to begin the management of a fleet of delivery vehicles.

pic: © https://memegenerator.net/instance/68420692/sarcastic-sloth-i-just-wanted-to-say-goodbye

[19] But much more important than universal quantum gates machines (IBM, Google etc.).
[20] *fr:* Le problème du voyageur de commerce.