

TP Optimisation – session 2 – Advent of Code

19 novembre 2024

Objectif d'apprentissage : cette séance vise à modéliser un problème issu de l'*advent of code* en programmation linéaire. Il s'agit du **jour 19 de l'année 2022**. Le scénario

Vous remarquez une collection de géodes¹ autour d'un étang rempli d'obsidiennes². Peut-être pourriez-vous utiliser l'obsidienne pour créer des robots craqueurs de géodes et les ouvrir ?

Pour collecter l'obsidienne au fond de l'étang, vous aurez besoin de robots collecteurs d'obsidienne étanches. Heureusement, il y a une grande quantité d'argile à proximité que vous pouvez utiliser pour les rendre imperméables.

Pour récolter l'argile, vous aurez besoin de robots collecteurs d'argile spéciaux. Pour fabriquer n'importe quel type de robot, vous aurez besoin de minerai, qui est également abondant, mais dans la direction opposée à celle de l'argile.

Pour collecter du minerai, il faut des robots collecteurs de minerai dotés de grosses foreuses. Heureusement, vous avez exactement un robot collecteur de minerai dans votre pack que vous pouvez utiliser pour lancer toute l'opération.

Chaque robot peut collecter une ressource de son type par minute. Il faut également une minute à l'usine de robots (qui se trouve également dans votre paquetage) pour construire n'importe quel type de robot, bien qu'elle consomme les ressources nécessaires disponibles au début de la construction.

L'usine de robots dispose de deux plans parmi lesquels vous pouvez choisir, mais une fois que vous l'avez configurée avec un plan, vous ne pouvez plus le modifier.

Par exemple :

Plan 1:

```
Chaque robot minéralier coûte 4 minerais.  
Chaque robot d'argile coûte 2 minerais.  
Chaque robot d'obsidienne coûte 3 minerais et 14 argiles.  
Chaque robot géode coûte 2 minerais et 7 obsidiennes.
```

Plan 2:

```
Chaque robot minéralier coûte 2 minerais.  
Chaque robot d'argile coûte 3 minerais.  
Chaque robot obsidienne coûte 3 minerais et 8 argiles.  
Chaque robot géode coûte 3 minerais et 12 obsidiennes.
```

Vous devez trouver le plan qui maximisera le nombre de géodes ouvertes au bout de 24 minutes en déterminant quels robots construire et à quel moment les construire.

En utilisant le plan 1 dans l'exemple ci-dessus, le plus grand nombre de géodes que vous pouvez ouvrir en 24 minutes est 9. Une façon d'y parvenir est :

`== Minute 1 ==`

```
1 robot collecteur de minerai collecte 1 minerai; vous avez maintenant 1 minerai.
```

`== Minute 2 ==`

```
1 robot ramasseur de minerai ramasse 1 minerai; vous avez maintenant 2 minerais.
```

1. Une cavité rocheuse tapissée de cristaux.

2. Une roche volcanique vitreuse et riche en silice.

== Minute 3 ==

Dépensez 2 minerais pour commencer à construire un robot collecteur d'argile.
1 robot collecteur de minerai collecte 1 minerai; vous avez maintenant 1 minerai.
Le nouveau robot collecteur d'argile est prêt; vous en avez maintenant 1.

== Minute 4 ==

1 robot collecteur de minerai collecte 1 minerai; vous avez maintenant 2 minerais.
1 robot ramasseur d'argile ramasse 1 argile; vous avez maintenant 1 argile.

== Minute 5 ==

Dépensez 2 minerais pour commencer à construire un robot collecteur d'argile.
1 robot collecteur de minerai collecte 1 minerai ; vous avez maintenant 1 minerai.
1 robot ramasseur d'argile ramasse 1 argile ; vous avez maintenant 2 argiles.
Le nouveau robot collecteur d'argile est prêt ; vous en avez maintenant 2.

== Minute 6 ==

1 robot collecteur de minerai collecte 1 minerai ; vous avez maintenant 2 minerais.
2 robots collecteurs d'argile collectent 2 argiles ; vous avez maintenant 4 argiles.

== Minute 7 ==

Dépensez 2 minerais pour commencer à construire un robot collecteur d'argile.
1 robot collecteur de minerai collecte 1 minerai ; vous avez maintenant 1 minerai.
2 robots collecteurs d'argile collectent 2 argiles ; vous avez maintenant 6 argiles.
Le nouveau robot collecteur d'argile est prêt ; vous en avez maintenant 3.

== Minute 8 ==

1 robot collecteur de minerai collecte 1 minerai ; vous avez maintenant 2 minerais.
3 robots collecteurs d'argile collectent 3 argiles ; vous avez maintenant 9 argiles.

== Minute 9 ==

1 robot ramasseur de minerai ramasse 1 minerai ; vous avez maintenant 3 minerais.
3 robots collecteurs d'argile collectent 3 argiles ; vous avez maintenant 12 argiles.

== Minute 10 ==

1 robot ramasseur de minerai ramasse 1 minerai ; vous avez maintenant 4 minerais.
3 robots collecteurs d'argile collectent 3 argiles ; vous avez maintenant 15 argiles.

== Minute 11 ==

Dépensez 3 minerais et 14 argiles pour commencer à construire un robot collecteur d'obsidienne.
1 robot collecteur de minerai collecte 1 minerai ; vous avez maintenant 2 minerais.
3 robots collecteurs d'argile collectent 3 argiles ; vous avez maintenant 4 argiles.
Le nouveau robot collecteur d'obsidienne est prêt ; vous en avez maintenant 1.

...

== Minute 21 ==

Dépensez 2 minerais et 7 obsidiennes pour commencer la construction d'un robot briseur de géodes.
1 robot collecteur de minerai collecte 1 minerai ; vous avez maintenant 3 minerais.
4 robots collecteurs d'argile collectent 4 argiles ; vous avez maintenant 29 argiles.
2 robots collecteurs d'obsidienne collectent 2 obsidiennes ; vous avez maintenant 2 obsidiennes.
1 robot briseur de géodes brise 1 géode ; vous avez maintenant 3 géodes ouvertes.
Le nouveau robot casseur de géodes est prêt ; vous en avez maintenant 2.

== Minute 22 ==

1 robot ramasseur de minerai collecte 1 minerai ; vous avez maintenant 4 minerais.
4 robots collecteurs d'argile collectent 4 argiles ; vous avez maintenant 33 argiles.

2 robots ramasseurs d'obsidienne ramassent 2 obsidiennes ; vous avez maintenant 4 obsidiennes.
 2 robots craqueurs de géodes cassent 2 géodes ; vous avez maintenant 5 géodes ouvertes.

== Minute 23 ==

1 robot collecteur de minerai ramasse 1 minerai ; vous avez maintenant 5 minerais.
 4 robots collecteurs d'argile collectent 4 argiles ; vous avez maintenant 37 argiles.
 2 robots ramasseurs d'obsidienne ramassent 2 obsidiennes ; vous avez maintenant 6 obsidiennes.
 2 robots craqueurs de géodes cassent 2 géodes ; vous avez maintenant 7 géodes ouvertes.

== Minute 24 ==

1 robot ramasseur de minerai ramasse 1 minerai ; vous avez maintenant 6 minerais.
 4 robots collecteurs d'argile collectent 4 argiles ; vous avez maintenant 41 argiles.
 2 robots ramasseurs d'obsidienne ramassent 2 obsidiennes ; vous avez maintenant 8 obsidiennes.
 2 robots craqueurs de géodes cassent 2 géodes ; vous avez maintenant 9 géodes ouvertes.

Cependant, en utilisant le plan 2 dans l'exemple ci-dessus, on souhaite savoir si on peut faire encore mieux.

Pour résoudre ce problème, il sera nécessaire d'utiliser des variables entières. Il ne s'agira donc pas d'un programme linéaire, mais d'un programme entier, ou programme linéaire en nombre entier (*Integer Linear Programming*). Dans ce cas, la résolution n'est plus polynomiale en temps mais s'il n'y a pas trop de variables entières, la résolution restera rapide.

Pour modéliser ce problème, il faut 4×24 variables binaires : une variable par type de robots que l'on décide de construire à chaque minute. On aura $x_{rm} \in \{0, 1\}$ pour $1 \leq r \leq 4$ (1 correspondant au robot minéralier, etc.) et $1 \leq m \leq 24$.

L'objectif consiste à maximiser le nombre de géodes ouvertes. Avec le plan 1, les deux robots de géodes ont été produits aux minutes 18 et 21. Ils ont donc permis d'ouvrir $(24 - 18) + (24 - 21) = 6 + 3 = 9$ géodes. Pour modéliser avec les variables binaires, on a $x_{4m} = 1$ pour $m \in \{18, 21\}$ et $x_{4m} = 0$ dans les autres cas. La formulation générale de l'objectif est donc la suivante :

$$\sum_{m=1}^{24} (24 - m) \times x_{4m}$$

Il faut maintenant rajouter les contraintes :

- Un seul robot construit à chaque minute. Par exemple à la minute 1, on aurait $x_{11} + x_{21} + x_{31} + x_{41} \leq 1$. En généralisant, on a $\sum_{r=1}^4 x_{rm} \leq 1$ pour $1 \leq m \leq 24$.
 - Suffisamment d'argile pour construire un robot obsidienne à chaque minute. Par exemple à la minute 3, on rajouterait l'argile déjà ramassé par les robots d'argile et on enlèverait l'argile qui aurait déjà été consommé pour construire des robots obsidiennes : $(3 - 1 - 1)x_{21} + (3 - 2 - 1)x_{22} - 14x_{31} - 14x_{32} \geq 14x_{33}$. En généralisant, on a $\sum_{m'=1}^{m-1} (m - m' - 1)x_{2m'} - bx_{3m'} \geq bx_{3m}$ pour $1 \leq m \leq 24$ avec $b = 14$.
 - Suffisamment d'obsidienne pour construire un robot géode à chaque minute.
 - Suffisamment de minerai pour construire n'importe quel robots à chaque minute.
- Sans les deux dernières contraintes, le programme entier est donc :

$$\begin{aligned} &\text{maximiser} && \sum_{m=1}^{24} (24 - m)x_{4m} \\ &\text{sous les contraintes} && \sum_{r=1}^4 x_{rm} \leq 1 \quad \text{pour } 1 \leq m \leq 24 \\ &&& \sum_{m'=1}^{m-1} (m - m' - 1)x_{2m'} - 8x_{3m'} \geq 8x_{3m} \quad \text{pour } 1 \leq m \leq 24 \\ &&& x_{rm} \in \{0, 1\} \quad \text{pour } 1 \leq r \leq 4 \text{ et } 1 \leq m \leq 24 \end{aligned}$$

Compléter avec les deux dernières contraintes.

maximiser
$$\sum_{m=1}^T (T - m)x_{4m}$$

sous les contraintes
$$\sum_{r=1}^4 x_{rm} \leq 1 \quad \text{pour } 1 \leq m \leq T$$

$$(m-1) + \sum_{m'=1}^{m-1} (m-m'-1)x_{1m'} - \sum_{r=1}^4 a_r x_{rm'} \geq \sum_{r=1}^4 a_r x_{rm} \quad \text{pour } 1 \leq m \leq T$$

$$\sum_{m'=1}^{m-1} (m-m'-1)x_{2m'} - bx_{3m'} \geq bx_{3m} \quad \text{pour } 1 \leq m \leq T$$

$$\sum_{m'=1}^{m-1} (m-m'-1)x_{3m'} - cx_{4m'} \geq cx_{4m} \quad \text{pour } 1 \leq m \leq T$$

$$x_{rm} \in \{0,1\} \quad \text{pour } 1 \leq r \leq 4 \text{ et } 1 \leq m \leq T$$

Où a_r est le coût en minerais du robot de type r , b le coût en argile du robot obsidienne, c le coût en obsidienne du robot géode et T est le nombre total de minutes.

On va pouvoir désormais coder ce programme entier pour résoudre le problème. Pour indiquer qu'une variable est entière, il faut l'indiquer à **pulp** (ici pour une variable booléenne) :

```
x = LpVariable("Variable booléenne", LpBinary)
```

Rajouter toutes les contraintes dans le code suivant, puis déterminer le plus grand nombre de géodes que vous pouvez ouvrir en 24 minutes ?

```
from pulp import *

a = [4, 2, 3, 2]
b = 14
c = 7
T = 24

vars = LpVariable.dicts("Robots", (range(4), range(T)), cat = LpBinary)

prob = LpProblem("Géodes", LpMaximize)
prob += lpSum((T - m - 1) * vars[3][m] for m in range(T)), "Géodes ouvertes"
for m in range(T):
    prob += lpSum(vars[r][m] for r in range(4)) <= 1
    prob += m + lpSum((m - mm - 1) * vars[0][mm] -
        lpSum(a[r] * vars[r][mm] for r in range(4))
        for mm in range(m)) >= lpSum(a[r] * vars[r][m] for r in range(4))

prob.solve()
print("Cout total = ", value(prob.objective))
```

```
from pulp import *

a = [2, 3, 3, 3]
b = 8
```

```

c = 12
T = 24

vars = LpVariable.dicts("Robots", (range(4), range(T)), cat = LpBinary)

prob = LpProblem("Géodes", LpMaximize)
prob += lpSum((T - m - 1) * vars[3][m] for m in range(T)), "Géodes ouvertes"
"
for m in range(T):
    prob += lpSum(vars[r][m] for r in range(4)) <= 1
    prob += m + lpSum((m - mm - 1) * vars[0][mm] -
        lpSum(a[r] * vars[r][mm] for r in range(4))
        for mm in range(m)) >= lpSum(a[r] * vars[r][m] for r in range(4))
    prob += lpSum((m - mm - 1) * vars[1][mm] - b * vars[2][mm]
        for mm in range(m)) >= b * vars[2][m]
    prob += lpSum((m - mm - 1) * vars[2][mm] - c * vars[3][mm]
        for mm in range(m)) >= c * vars[3][m]

prob.solve()
print("Cout total = ", value(prob.objective))

```

On obtient 12.