Rapport de projet Hashcode 2017

T. Gomez, P. Li, N. Maraval, M. Monvoisin, L. Pagano

12 janvier 2018

1 Architecture

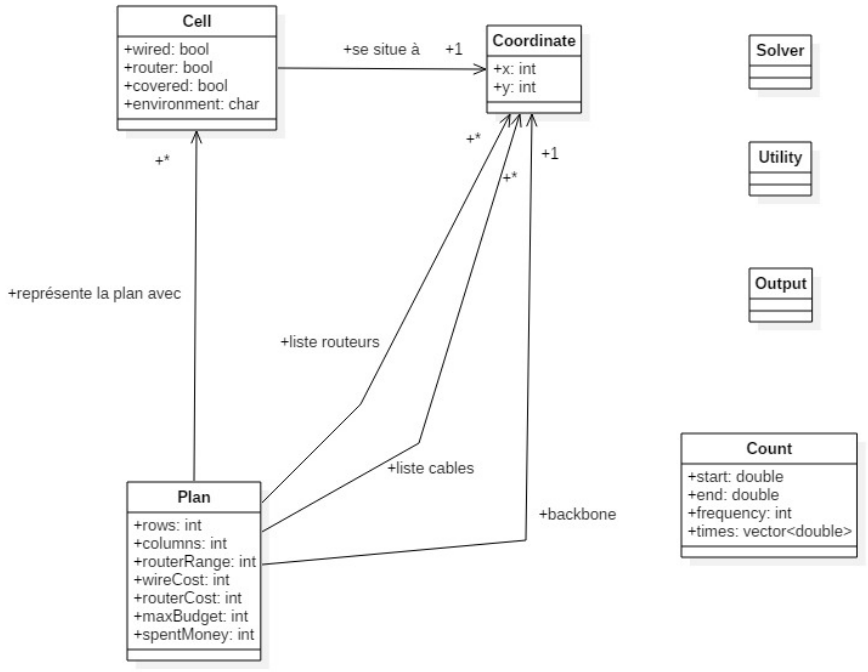

FIGURE 1 – Diagramme de classe

Dans notre architecture, c'est la classe **Plan** qui va représenter l'état de la map, avec le placement des routeurs et des câbles. Cette classe contient aussi les informations liées à la map, comme le budget, le coût des routeurs, ou encore la portée des routeurs.

Chaque case de la map est modélisée par une instance de **Cell**. Chaque cellule va porter les informations avec des booléens pour savoir s'il y a un routeur sur la case, ou encore du wifi. De plus, la cellule possède une instance de **Coordinate**, afin d'accéder aux coordonnées directement depuis la cellule. La classe **Coordinate** va aussi nous permettre de faire tous les calculs de distance entre les cases.

La classe **Solver** permet de placer tous les routeurs sur une instance de **Plan**. Pour cela, il va s'aider des méthodes de la classe **Utility**, qui possède toutes les méthodes générales de notre code. Enfin, c'est la classe **Output** qui va produire le fichier d'ouptut à partir de l'instance de **Plan**.

La classe **Count** permet de mesurer le temps d'exécution du programme. Evidemment, il va faire une moyenne sur un nombre d'essais paramétrable avec l'attribut frequency.

2 Arbitre

L'arbitre prend deux paramètres ; le nom du répertoire et le nom du fichier de sortie. Le nom du fichier de sortie est de la forme nomMap.out, le nom de la map en est donc déduit. Cela permet

de charger les informations correspondant à cette map, en utilisant la classe Plan avec le fichier d'entrée nomMap.in.

Grâce à cela, l'arbitre vérifie ensuite les points suivants :

— chaque routeur n'est ni sur un mur ni une cellule vide
— le budget maximal n'est pas dépassé
— tous les routeurs sont reliés par un câble au backbone
— toutes les cellules sont connectées au backbone ou à la cellule précédente

3 Stratégie

3.1 Placement des routeurs

On note r_M la portée des routeurs sur la carte M. Le nombre de case maximum qu'un routeur peut recouvrir est alors $N_M = (2r_M + 1)^2$.

Pour placer les routeurs, chaque case de la carte est parcourue de gauche à droite et de haut en bas pour trouver des emplacements de routeurs qui permettent chacun de couvrir N_M cases non déjà couvertes. L'algorithme 1 permet de placer les routeurs. On définit une fonction $getCoverableCells(i, j)$ qui renvoit les cases qui peuvent être couvertes par un routeur placé sur la case (i, j) et qui ne sont pas déjà couvertes. La condition pour placer un routeur sur une case (i, j) est la suivante :

$$length(getCoverableCells(i, j)) = N_M \tag{1}$$

Algorithm 1 Placer routeurs

```
Input :
— M                                                    ▷ La carte
for i = 0 to lignes(M) do
    for j = 0 to colonnes(M) do
        if length(getCoverableCells(i, j)) = N_M then
            addRouter(i,j)
        end if
    end for
end for
```

3.2 Câblage

La méthode pour cabler les routeurs consiste à trouver la paire de routeurs (r_1, r_2) telle que :

$$(r_1, r_2) = \underset{\substack{r'_1 \in NonCab \\ r'_2 \in Cab}}{argmin} \; dist_{oct}(r'_1, r'_2) \tag{2}$$

Où $dist_{oct}$ est la distance octogonale, $NonCab$ est l'ensemble des routeurs non cablés et Cab est l'ensemble des routeurs câblés. Une fois que cette paire de routeurs est trouvée, il faut la câbler. Il est possible de câbler directement ces deux routeurs en cherchant un chemin de longueur minimale, mais il existe beaucoup de chemins différents de longueur minimale, et en fonction de celui choisi, des câbles pourraient être rajoutés inutilement. La méthode $followWire$ permet de remédier partiellement à cela.

3.2.1 Méthode followWire

Si nous voulons câbler un routeur nous allons donc d'abord trouver le routeur le plus proche avec *linkTwoGroups*. Le routeur de départ est déjà câblé, et le second ne l'est pas encore. Il est alors possible que le routeur de départ soit déjà relié à un cable qui est plus proche de la destination. Dans cette optique la méthode *followWire* va suivre, si possible, le chemin du cable tant qu'il se rapproche du second routeur. La méthode va donc renvoyer la case câblée la plus proche du routeur de destination.

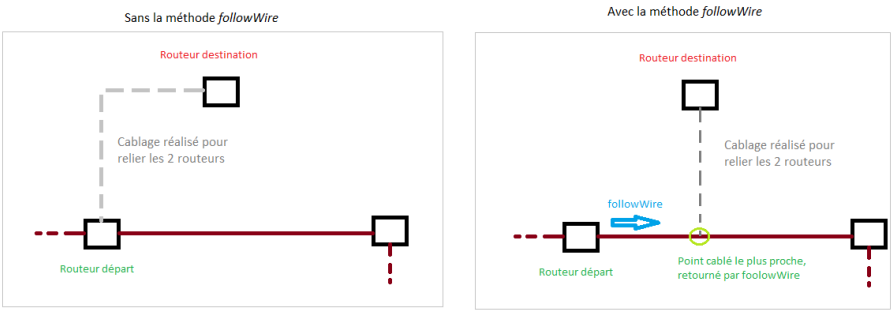

FIGURE 2 – Illustration de la méthode followWire

3.2.2 Algorithme

La fonction $link(a, b)$ câble toutes les cases situées entre la cellule a et la cellule b en passant par le plus court chemin (en utilisant la distance octogonale). L'algorithme commence par cabler la cellule a. Si une case du chemin est déjà câblée, on efface les câbles posés jusqu'à présent et on reprend le cablage à partir de cette case. Cette méthode ne renvoie rien.

Algorithm 2 câbler routeurs

Input :
— M ▷ La carte
— L_{nonCab} ▷ La liste des routeurs déjà placés
$L_{cab} \leftarrow \{backBone\}$
while $L_{nonCab} \neq \varnothing$ **do**
  $r_{nonCab},\ r_{cab} \leftarrow linkTwoGroups(L_{nonCab}, L_{cab})$
  $c_{cab} \leftarrow followWire(r_{nonCab},\ r_{cab})$
  $link(r_{nonCab}, c_{cab})$
  $L_{nonCab} \leftarrow L_{nonCab} \setminus r_{nonCab}$

end while

L'algorithme 2 permet de câbler les routeurs.

3.3 Complétion des trous

Une fois que les routeurs couvrant la plus grande aire possible ont été placés et câblés, d'autres routeurs couvrant une aire moindre peuvent être placés et câblés. Pour cela, les cases de la carte M sont reparcourues de gauche à droite et de haut en bas comme lors de la première phase, où la condition pour poser un routeur sur la case (i, j) devient :

$$length(getCoverableCells(i, j)) = N \tag{3}$$

Où $N < N_M$. La valeur de N est initialisé à $N_M - 1$. Une fois que le parcours de la carte est terminé, il reprend avec une valeur de N décrémentée de 1. Lorsque N atteint 0, l'algorithme s'arrête. L'algorithme 3 permet de placer d'autres routeurs sur la carte.

Algorithm 3 Complétion des trous

Input :
— M ▷ La carte
— L_{cab} ▷ La liste des routeurs déjà câblés
$N \leftarrow N_M -$
while N $>$ 0 and budgetNonDepasse() **do**
  for $i = 0$ **to** $lignes(M)$ **do**
    for $j = 0$ **to** $colonnes(M)$ **do**
      if $length(getCoverableCells(i, j)) = N$ **then**
        $r \leftarrow addRouter(i, j)$
        $\_,\ r_{cab} \leftarrow linkTwoGroups(\{r\}, L_{cab})$
        $c_{cab} \leftarrow followWire(r, r_{cab})$
        $link(r, c_{cab})$
      end if
    end for
  end for
  $N \leftarrow N - 1$
end while