SI J'AVAIS UNE SCIE

Année 2015-2016

Élèves :CLERC Max 1S2, GEROSA Eva 1S4, MARTIN Cécilia 2I2, KEMPF Anaïs 1M2, MÉDOC Antoine 1S2, OHANA Arthur 1S4, SARR Khadijatou 1S2, SCHOEN Alizée 1S2.

Encadré.e.s par : BIGLIONE Sabrina, JACQUIER Murièle, LAMBOLEY Gilles

Chercheurs : BENHEDDI Mounir, Université de Genève et MACLEAN Catriona, Institut Joseph Fourier à Grenoble

Établissements : Lycée de La Versoie - Thonon les Bains, Institut Florimont - Petit Lancy (Suisse)

Présentation du sujet

Une personne souhaite paver sa terrasse. Pour ce faire, elle dispose de pavés rectangulaires uniformes. Cependant, sa terrasse contient des arbres qui l'empêchent de savoir si elle pourra y arriver.

On modélise la terrasse par une grille rectangulaire composée de cases carrées adjacentes et de même taille que l'on va représenter sous forme de damier. Les arbres recouvrent une case et les pavés recouvrent deux cases adjacentes. La grille est considérée comme complète lorsque toutes les cases de la grille sont recouvertes par un arbre ou un pavé.

Nous nous sommes fixé deux objectifs :
- savoir dans quel cas il sera possible de paver une grille
- réussir à paver une grille possible (grille qu'il est possible de paver)

Annonce des conjectures et résultats obtenus

Suite à nos recherches, nous sommes parvenus à réaliser un algorithme indiquant si une grille est sensiblement possible ou non sans la remplir. Cependant, il existe un contre-exemple à cet algorithme.

Nous avons également réalisé un algorithme que nous avons supposé capable de résoudre n'importe quelle grille possible et, si ce n’est pas le cas, d'indiquer que la grille est impossible.

Texte de l’article

I - Variables de base :

Nous avons défini quelques premières variables :

a	nombre d'arbres sur la grille
x	longueur de la terrasse
y	largeur de la terrasse
$A = x \times y$	surface de la grille
$t = (A-a)/2$	nombre de pavés nécessaires pour compléter la grille

Une unité équivaut au côté d’une case.

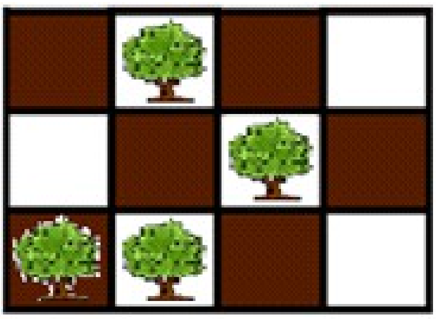
II - Quelques Notions

Pour nous aider à résoudre ce problème, nous avons dû intégrer 3 particularités à nos grilles. Il est nécessaire de les comprendre pour pouvoir suivre le reste de l’article.

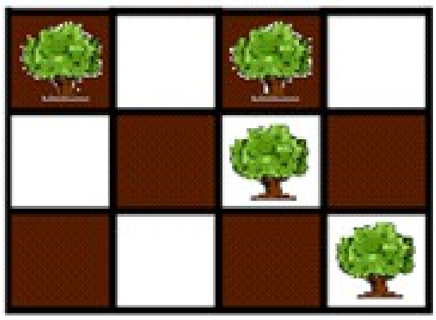
1) Les zones

D’abord, nous avons délimité des zones dans chaque grille à paver. Celles-ci sont délimitées par les bords de la grille et les arbres. Une zone peut être « paire » ou « impaire », selon le nombre de cases qui la composent.

Il est à noter qu’une grille aléatoire est généralement composée d’une zone unique.



Deux zones impaires



Deux zones paires

figure n°1 : Zones paires et zones impaires

2) Cases Blanches et Cases Noires

Ensuite, les cases de la grille sont différenciées par deux couleurs : blanc et noir. Les cases d’une même couleur se placent en diagonales les unes des autres. Chaque case aura donc pour voisines directes 4 cases de couleur différente de la sienne.

Cette répartition en deux couleurs est identique à celle d’un damier ou d’un échiquier.

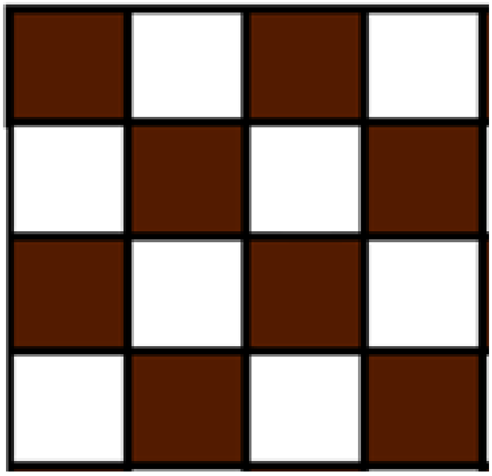


Figure n° 2 : Grille avec les cases blanches et noires.

La troisième, et dernière notion, est un peu plus compliquée que ses sœurs.
Nous avons associé à chaque case « pavable » d'une grille « un nombre de fuite », compris entre 1 et 4. Celui-ci est égal au nombre n de cases pavables adjacentes (sans les diagonales) à la case et est noté F=n. L'image ci-dessous montre le nombre de fuite de chaque case d'une grille.

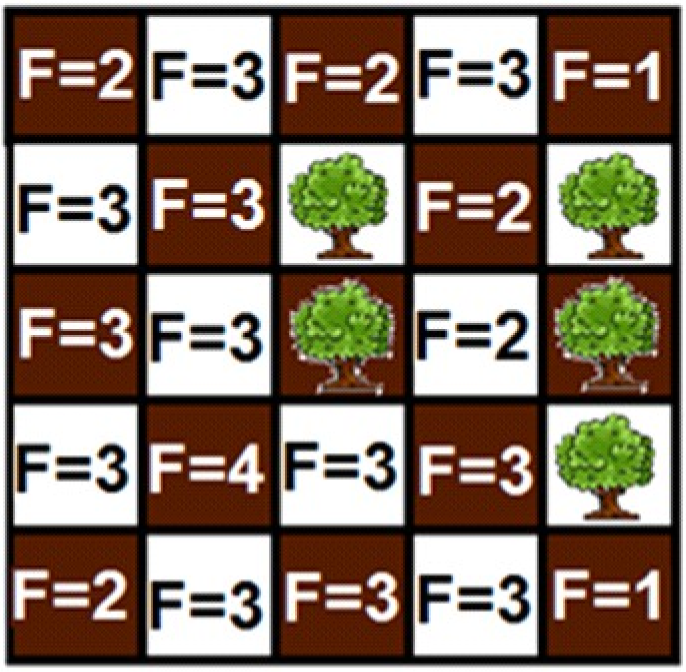


Figure n°3 : Grille avec le nombre de fuite par case

Lorsqu'une case est « F=0 », c'est qu'elle n'a pas de voisine. Il sera alors impossible de mettre un pavé sur cette case et on ne pourra donc pas paver la grille.

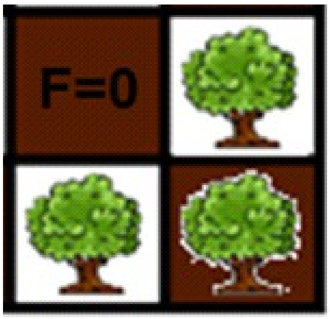


Figure n°4 : Exemple de case avec un nombre de fuite nul (« F=0 »).

Lorsqu'une case est « F=1 », on dit qu'il s'agit d'une priorité. Nous devons alors placer un pavé sur cette case et sa seule possibilité de fuite. Une autre option créerait une case isolée avec F=0 et aboutirait à une grille impossible à paver.

Dans certains cas, comme ci-dessous (figure n°5) les priorités se chevauchent, ce qui rend la grille impossible à paver.

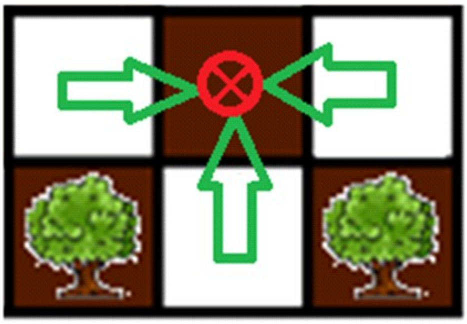


Figure n°5 : Chevauchement de priorités

On peut vérifier si des priorités se chevauchent si on a un cas (F=1) +(F=2)+(F=1)

Lorsque deux « F=2 » adjacents ont leurs deuxièmes fuites également adjacentes, on les relie entre elles (voir ci-dessous), car cette solution n'enlève aucune option de résolution. Nous avons appelé cette disposition un « bol ».

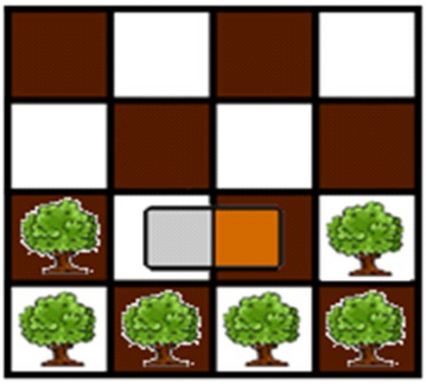


Figure n°6 : Exemple d'un bol

III - Savoir si une grille est possible

Nous avons tout d'abord cherché à savoir, à l'avance, si une grille sera possible (on peut la paver) ou non. Nous avons finalement trouvé un algorithme en 3 étapes.

1) Zones paires et zones impaires

La première étape consiste à vérifier que chaque zone de la grille a un nombre pair de cases à paver. Si ce n'est pas le cas, on peut aisément en conclure que la grille ne pourra pas être remplie, un pavé occupant 2 cases.

2) Les priorités

La deuxième étape consiste à placer toutes les priorités, en vérifiant qu'elles n'en créent pas de nouvelles. Si c'est le cas, il faut les placer à leur tour et ainsi de suite. Si un chevauchement de priorités ou un F=0 venait à être créé durant cette étape, ça signifie que la grille ne pourra pas être pavée, seuls les pavés strictement obligatoires étant posés.

3) Cases blanches, cases noires

La dernière étape consiste à vérifier que la grille a autant de cases blanches que de cases noires à paver. Si ce n'est pas le cas, la grille ne peut pas être entièrement pavée car un pavé repose nécessairement sur une case blanche et une case noire. Sinon, il est possible de paver la grille. (1)
Ces étapes peuvent se condenser dans le schéma suivant :

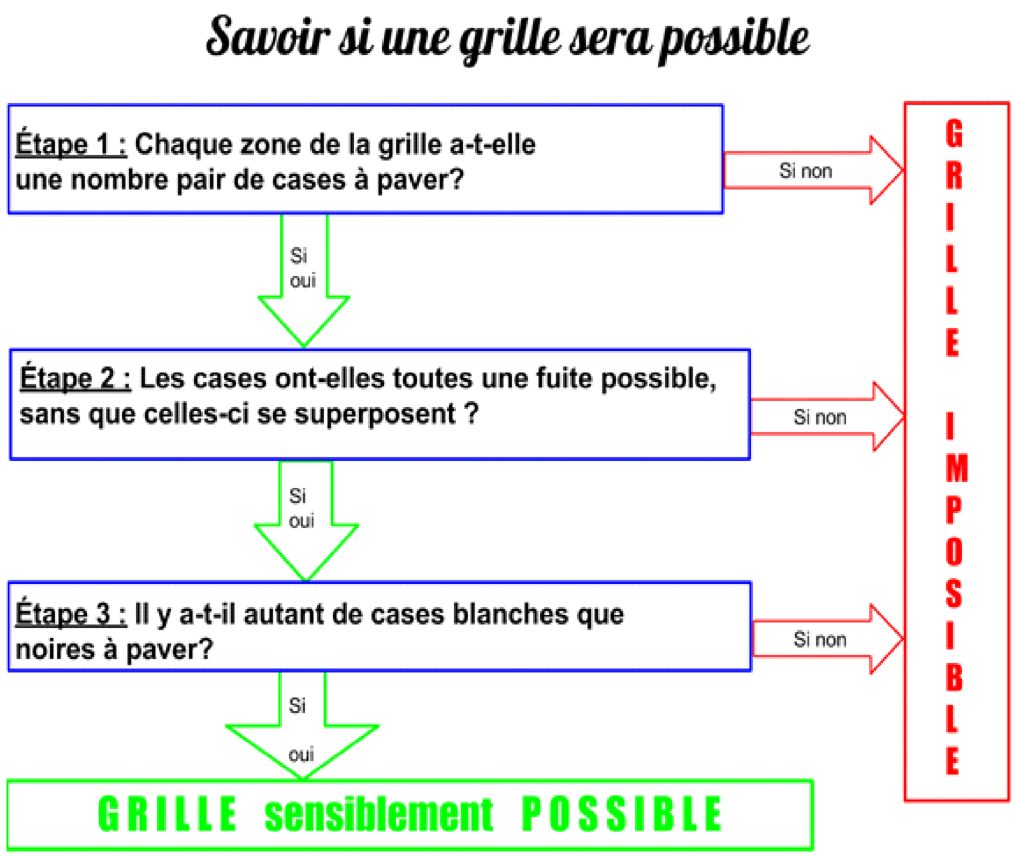


Figure n°7 : Il est à noter que cet algorithme n'est que le fruit de conjectures. On peut admettre qu'il sera juste en donnant une grille impossible à paver, ces étapes étant issues d'une logique générale. Cependant, il existe des contre-exemples où des grilles sont considérées comme possibles alors qu'elles ne peuvent pas être pavées. C'est cependant le meilleur que nous sommes parvenus à faire sans poser de pavé.

IV - Résoudre une grille

1) Introduction

Afin de résoudre une grille possible, nous avons élaboré une méthode présentée ici sous forme d'un algorithme. Celle-ci se base sur le système des priorités détaillé précédemment.

En effet, on cherche à compléter progressivement la grille, en plaçant des pavés qui ne la rendraient pas impossible.

2) Présentation de l'algorithme

Soit une grille avec un nombre arbres :

Etape 1 - Si la grille est complète, il n'y a plus de cellules vides, l'objectif est atteint.

La grille est-elle complète (*f=0* pour chaque cellule de la grille) ? (2)

- Si oui : FIN de l'algorithme, grille résolue.
- Si non : aller à l'étape 2.

Etape 2 - Les cellules avec *f=1* sont les premières à être complétées car par définition il existe *n* façons de compléter une cellule avec *f=n*. Aussi il est obligé de placer un pavé sur les cellules avec *f=1*.

Existe-t-il des obligations (cellules avec *f=1*) ?

- Si oui : en compléter une. Quatre issues possibles après cela :
- la grille est complète : FIN de l'algorithme, grille résolue.
- une ou plusieurs cellules sont passées à *f=0* : FIN de l'algorithme, grille impossible.
- il existe d'autre cellules avec *f=1* ou bien une ou plusieurs cellules sont passées à *f=1* : aller à l'étape 2.

Aller à l'étape 1, après avoir complété une cellule avec *f=1* permet d'envisager toutes ces possibilités.

- Si non : passer à l'étape 3.

Etape 3 - Les formes de bols définies précédemment font partie d'un groupe de "configurations prioritaires" qui peuvent être complétées, sans pour autant rendre une grille réalisable impossible, en bloquant certaines solutions.

En effet, on remarque qu'il n'existe que 5 façons de compléter ce rectangle et d'en "sortir".

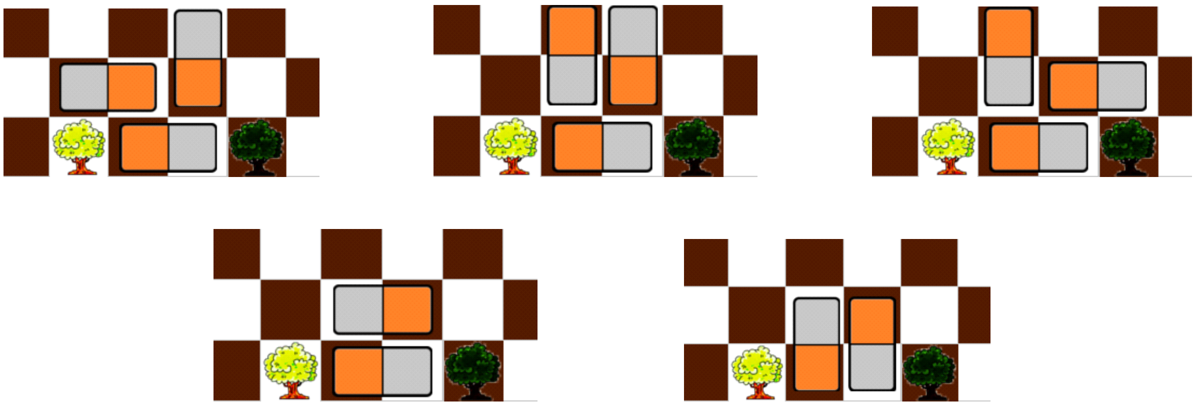


Figure n°8 : Les 5 façons de paver un bol

Or, toutes sont accessibles en complétant les formes de bols comme indiqué précédemment :

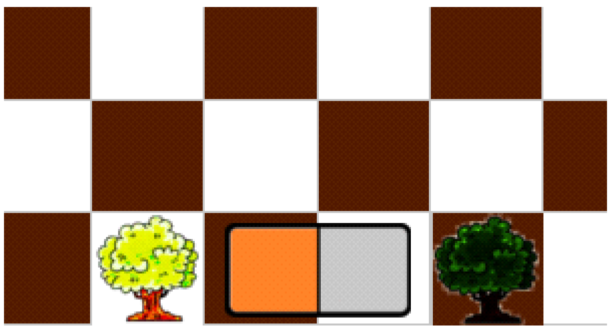


Figure n°9 :

On complète ainsi la grille sans pour autant la priver de solution (même si cette action peut limiter le nombre de solutions de la grille).

Existe-t-il des formes de bols ?

- Si oui : en compléter une. Deux issues possibles après cela :
- une ou plusieurs cellules sont passées à *f=1* : aller à l'étape 2.
- il existe d'autres formes de bols ou une ou plusieurs formes de bols se sont formées : aller à l'étape 3.

Aller à l'étape 2 après avoir complété une forme de bol permet d'envisager toutes ces possibilités.

- Si non, passer à l'étape 4.

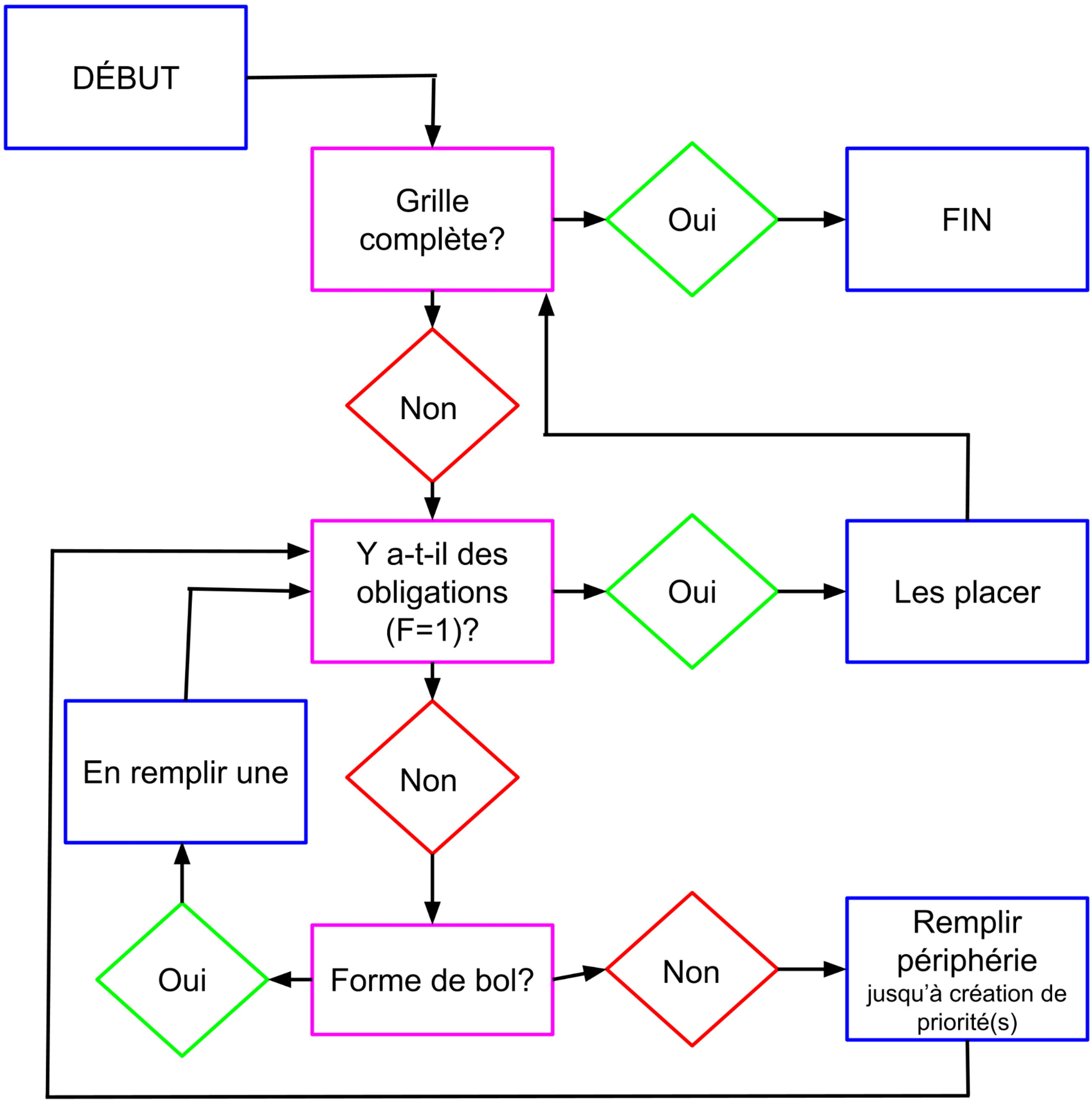
Etape 4 - Lorsqu'il n'y a ni cellule à *f=1,* ni forme de bol et que la grille est toujours incomplète mais potentiellement possible avec les informations dont nous disposons, alors il faut compléter la périphérie depuis un coin le plus proche du bord de la grille, jusqu'à la formation d'une priorité (*f=1* ou bol). Faire ceci, permet de réduire la surface de travail pour simplifier la résolution.

Cette étape n'est pas démontrée, il s'agit d'une conjecture forte que nous avons. Après de très nombreux essais nous n'avons pas trouvé de contre-exemple. En plus de l'absence de justification mathématique se pose le problème du choix du coin pour démarrer le remplissage de la périphérie (doit-il finalement être fait au hasard, si oui pourquoi ?).

- Compléter la périphérie jusqu'à création d'une nouvelle priorité (cellule(s) avec *f=1* ou forme(s) de bol). Deux issues possibles après cela :
- une ou plusieurs cellules sont passées à *f=1* : aller à l'étape 2.
- une ou plusieurs formes de bols se sont formées : aller à l'étape 3.

Aller à l'étape 2 après avoir complété la périphérie permet d'envisager toutes ces possibilités.

Nous avons donc représenté cette méthode sous forme d'un schéma de l'algorithme le représentant, dans l'hypothèse d'une grille possible :



Tel que représenté, l'algorithme ne donne pas toutes les résolutions possibles et suppose une application dans une grille possible ; en effet on ne vérifie pas les cellules à $f=0$.

Conclusion de la troisième partie

Le fait que cet algorithme puisse résoudre toutes les grilles possibles est donc une conjecture forte. Il reste toujours une étape non démontrée.

On peut appliquer cette méthode après l'algorithme présenté précédemment qui peut éliminer certaines grilles dont on a la certitude qu'elle sont impossibles. Il ne donnera pas toutes les résolutions possibles.

Il existe deux points de cette partie sur lesquels nos recherches restent inachevées :

- Configurations prioritaires :

On trouve parmi elles les bols ou les carrés prioritaires évoqués précédemment. Il s'agit de l'ensemble des configurations locales que l'on peut compléter d'une manière à ne pas priver une grille possible de solutions.
Il en existe possiblement une infinité et elles auraient pu être incluses dans un algorithme de résolution plus complet. Nous pourrions même envisager qu'en trouver un très grand nombre pourrait permettre de supprimer l'étape 4 restant sans démonstration.

- Zones abstraites :

Étudiées dans le but de remplacer l'étape 4, elles sont définies comme tous les rectangles les plus grands possibles pouvant être tracés dans une grille et aucun d'eux ne comprend ni obstacles ni d'autres rectangles en entier.
La délimitation de la surface de travail en zones abstraites nous aide à remplir une grille sans recourir à l'étape 4. Elle était plus propice à une démonstration mathématique. Nous n'avons pas approfondi cette piste.

Conclusion

Notre premier problème, à savoir identifier une grille comme possible ou impossible sans placer aucun pavé, n'est donc pas entièrement résolu par l'algorithme proposé car il existe des contre-exemples et la démonstration mathématique de nos différentes conjectures s'annonce difficile.
Malgré cela notre algorithme s'avère efficace dans une majorité des cas testés.

Par ailleurs, nous avons réussi à développer une méthode (algorithme de résolution partie IV) pour remplir une grille possible et, si elle ne l'est pas, le savoir. Cet algorithme fonctionne dans tous les cas testés. Malgré tout, une de nos étapes n'a pas été démontrée et son efficacité reste une conjecture forte.

Annexe

Calcul du nombre de cases noires et blanches dans une grille impaire ou paire

Soit une grille rectangulaire de dimension $(x;y)$ avec les cases repérées par $(k;l)$ où k *est* leur numéro de colonne et l leur numéro de ligne en prenant la case en bas à gauche aux coordonnées $(1;1)$. La case directement au dessus sera donc repérée par $(1;2)$ et celle directement au droite par $(2;1)$.

De plus $k \in N$ et $k \in [1;x]$
$l \in N$ et $l \in [1;y]$

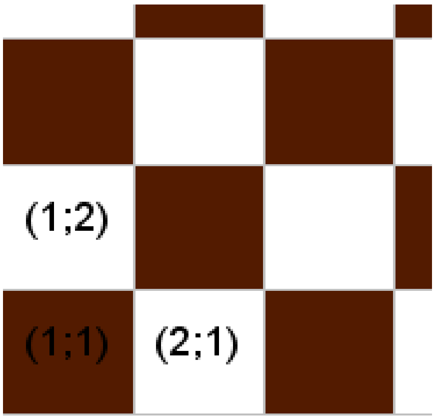


figure n°10

Soit *N* le nombre de cases noires et *B* le nombre de cases blanches, qui sont disposées comme sur un échiquier, en plaçant au moins une case noire dans un des coins : sur les points *(1;1)* ou *(x;1)* ou *(1;y)* ou *(x;y)*.

Soit *n* un entier naturel.

1) Si x est impair, alors on peut définir deux types de lignes différents :

En effet deux types de lignes s'alternent et deux types de colonnes s'alternent, cela étant dû aux cases noires et aux cases blanches.

<u>1er type de ligne :</u> Ce premier type de ligne intègre toutes les cases repérées par *(k;2n+1)* avec 2n+1 ≤ y.

On étudie un cas particulier :

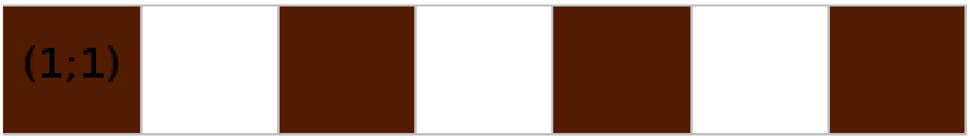


figure n°11

Comme la première case est noire aux coordonnées *(1;1)* et que l'on alterne cases blanches et cases noires, alors toutes les cases repérées par *(2n+1;1)* seront noires également et les cases repérées par *(2n;1)* seront blanches en excluant *(0;1).*

On donne *N'* et *B'* respectivement le nombre de cases noires et de cases blanches jusqu'à la colonne *x-1*. Comme x est impair alors *x-1* est pair et *(1;x-1)* repère une case blanche. (3) On a donc :

B' = (x-1)/2

B' = 0,5x - 0,5

Or comme (1;1) repère une case noire alors on a :

B' = N'

N' = 0,5x – 0,5

Or comme x est impair alors il est de forme 2n+1 et (1;x) repère une case noire. Soit B et N respectivement le nombre total de cases blanches et de cases noires sur cette ligne :

B' = B

B = 0,5x - 0,5

N' = N - 1

N = N'+1

N = 0,5x - 0,5 + 1

N = 0,5x + 0,5

<u>2e type de ligne :</u> Ce deuxième type de ligne intègre toutes les cases repérées par *(k;2n)* avec *2n∈[1;y].*

On étudie un cas particulier :

figure n°12

Comme la première case est blanche aux coordonnées *(2;1)* et que l'on alterne cases blanches et cases noires alors toutes les cases repérées par *(2n+1;1)* seront blanches également et les cases repérées par *(2n;1)* seront noires en excluant (0;1).

Soit B et N respectivement le nombre total de cases blanches et de cases noires sur cette ligne. De même que précédemment on arrive à :

B = 0,5x + 0,5

N = 0,5x - 0,5

2) Si x est pair, alors on peut définir deux types de lignes différents :

<u>1er type de ligne :</u> Ce premier type de ligne intègre toutes les cases repérées par *(k;2n+1)* avec *2n+1 ≤ y.*

On étudie un cas particulier :

figure n°13

Comme la première case est noire aux coordonnées *(1;1)* et que l'on alterne cases blanches et cases noires alors toutes les cases repérées par *(2n+1;1)* seront noires également et les cases repérées par *(2n;1)* seront blanches en excluant (0;1)

On donne *N* et *B* respectivement le nombre total de cases noires et de cases blanches sur cette ligne.
Comme x est pair alors *(1;x)* repère une case blanche. On a donc :

B = x/2

B = 0,5x

Or comme (1;1) repère une case noire alors on a :

B = N
N = 0,5x

<u>2e type de ligne :</u> Ce deuxième type de ligne intègre toutes les cases repérées par *(k;2n)* avec $2n \in [1;y]$.
On étudie un cas particulier :

figure n°14

Comme la première case est blanche aux coordonnées *(2;1)* et que l'on alterne cases blanches et cases noires alors toutes les cases repérées par *(2n+1;1)* seront blanches également et les cases repérées par *(2n;1)* seront noires en excluant (0;1).

Soit B et N respectivement le nombre totale de cases blanches et de cases noires sur cette ligne. De même que précédemment on arrive à :
B = 0,5x
N = 0,5x

3) Si y est impair, alors on peut définir deux types différents de colonnes :

<u>1er type de colonne :</u> Ce premier type de colonne intègre toutes les cases repérées par *(2n+1;l)* avec $2n+1 \leq x$.
On étudie un cas particulier :

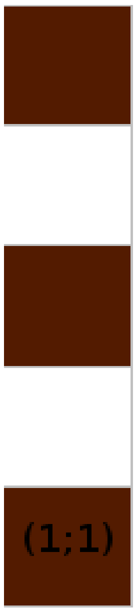


figure n° 15

Soit B et N respectivement le nombre total de cases blanches et de cases noires sur cette colonne.
De la même façon que pour les lignes on arrive à :

B = 0,5y - 0,5
N = 0,5y + 0,5

<u>2e type de colonne :</u> Ce deuxième type de colonne intègre toutes les cases repérées par *(2n;l)* avec *2n∈[1;x].*

Elles ne sont pas utiles pour notre démonstration.

4) Si y est pair, alors on peut définir deux types de colonnes différents :

<u>1er type de colonne :</u> Ce premier type de colonne intègre toutes les cases repérées par *(2n+1;l)* avec *2n+1 ≤ x*

On étudie un cas particulier :

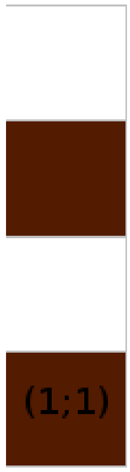


figure n°16

Soit B et N respectivement le nombre total de cases blanches et de cases noires sur cette colonne. De la même façon que pour les lignes on arrive à :

B = 0,5y
N = 0,5y

<u>2e type de colonne :</u> Ce deuxième type de colonne intègre toutes les cases repérées par *(2n;l)* avec *2n∈[1;x]*
Elles ne sont pas utiles pour notre démonstration.

En déduire le nombre de cases blanches et des cases noires pour chaque type de grille :

En effet, on remarque que sur une grille le nombre de cases noires d'une colonne de type 1 correspond au nombre de lignes de type 1 tandis que le nombre de cases blanches d'une colonne de type 1 correspond au nombre de lignes de type 2.

5) En déduire le nombre de cases blanches et des cases noires pour chaque type de grille :

On note B et N respectivement le nombre total de cases blanches et de cases noires d'une grille.

	x pair	*x* impair

y pair	Grille paire avec : *B =* *B =* *N + B = xy = A* *N - B = 0*	Grille paire avec : *N = 0,5y(0,5x + 0,5) + 0,5y(0,5x - 0,5)* *N = 0,5xy* *B = 0,5y(0,5x - 0,5) + 0,5y(0,5x + 0,5)* *B = 0,5xy* *N + B = xy = A* *N - B = 0*
y impair	Grille paire avec : *N=* *B = 0,5xy* *B =* *B = 0,5xy* *N + B = xy = A* *N - B = 0*	Grille impaire avec : *N = (0,5y + 0,5)(0,5x + 0,5) + (0,5y - 0,5)(0,5x - 0,5)* *N = 0,5xy + 0,5* *B = (0,5y + 0,5)(0,5x - 0,5) + (0,5y - 0,5)(0,5x + 0,5)* *B = 0,5xy - 0,5* *N + B = xy = A* *N - B = 1*

On remarque ainsi que pour toutes grilles paires on a autant de cases blanches que de cases noires alors que pour toutes grilles impaires on a 1 case noire de plus que de cases blanches.

Algorithme sur Minecraft

Nous avons réalisé un algorithme sur Minecraft que vous pouvez télécharger ici : http://www.planetminecraft.com/project/math-en-jeans-if-i-have-a-saw-school-projectHYPERLINK "http://www.planetminecraft.com/project/math-en-jeans-if-i-have-a-saw-school-project-/"-/

Notes d'édition

(1) Le nombre de cases noires et blanches est expliqué dans l'annexe.

(2) Dans cette partie, F est noté *f.*

(3) Il y a échange entre numéro de ligne et numéro de colonne. Sur la première ligne, la case correspondant à la colonne x-1 a pour coordonnées (x-1 ; 1) et non (1 ; x-1). Le même échange se produit jusqu'à la fin de la partie 4) .