

# Rassemblement d'agents mobiles

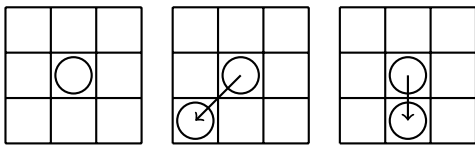
Éloi Perdereau

24 avril 2014

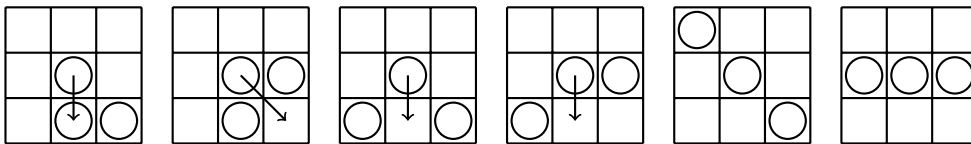
## 1 Cas

Les cas symétriques sont omis.

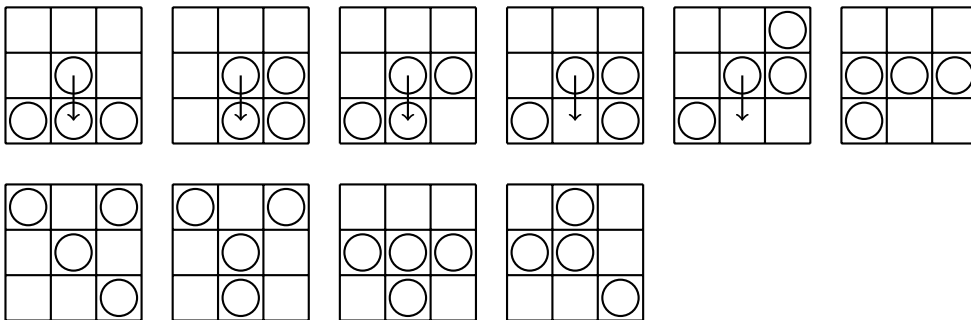
### 1.1 0 ou 1 voisins



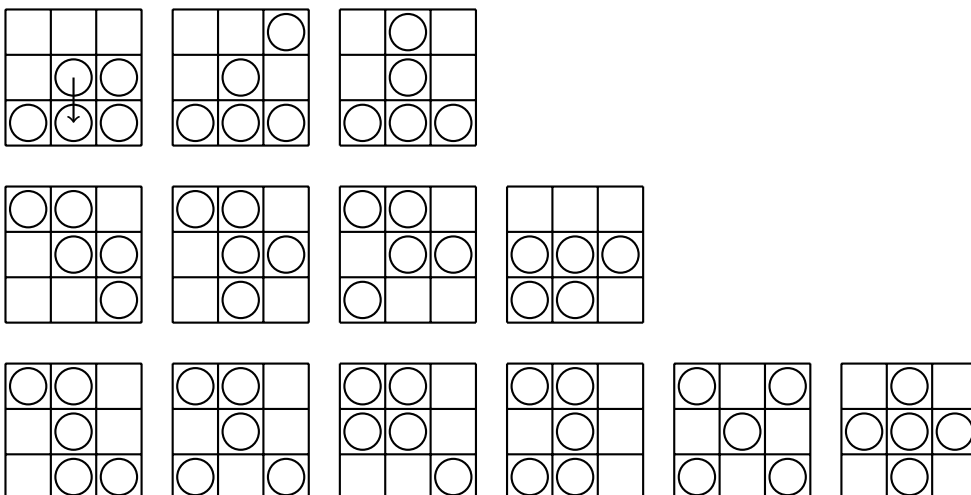
### 1.2 2 voisins



### 1.3 3 voisins



### 1.4 4 voisins



## 1.5 5 voisins et plus

Pour énumérer les cas avec 5 et 6 robots voisins, il suffit de prendre le complémentaire des cas avec respectivement 3 et 2 robots voisins. Aucun de ces cas n'entraîne un mouvement de la part du robot concerné.

## 1.6 Extension

Tel quels, les cas 6 et 13 peuvent conduire à une déconnexion de l'espace. Pour y remédier, il faut que chaque robot mémorise son entourage d'une ronde sur l'autre (il ne retient que son entourage précédent.) Puis, si à la ronde précédente, il était dans le cas 6 ou 13, il faut qu'il vérifie si au moins une des cases suivante contient un robot : à droite, en bas et en bas à droite. Si ce n'est pas le cas, il revient à sa position précédente. Les cas symétriques sont définis de façon analogues.

Après avoir réglé ces cas de déconnexions, un autre problème survient avec les cas 5 et 6. Il se peut que l'espace alterne entre deux états ce qui rend le rassemblement impossible. Le problème vient du fait que des robots disposés en quinconce soient de nouveau en quinconce à la ronde suivante (avec des positions inversés.) Et ainsi, revenir à la position qu'ils occupaient deux rondes plus tôt. Pour y remédier, on va de nouveau utiliser l'entourage de la ronde précédente : Si à la ronde précédente, un robot était dans le cas 5 ou 6, et qu'il est désormais dans le cas opposé, alors il ne bouge pas pour cette ronde.

## 1.7 Formalisation

---

**Algorithm 1 :**

---

```
finish  $\leftarrow$  False;
ok  $\leftarrow$  True;
k  $\leftarrow$  0;
while not finish do
    Nk  $\leftarrow$  get_neighbors();
    if k%4 = 0 then
        if Nk is case 1.2.4 or 1.3.5 then
            move to (i, j - 1);
            get_neighbors();
            if (i, j - 2) or (i + 1, j - 2) is not empty then
                ok  $\leftarrow$  False;
            end
        end
    else if k%4 = 1 then
        if Nk-1 is case 1.2.4 or 1.3.5 then
            move to (i, j + 1);
        end
    else if k%4 = 2 then
        if Nk is case 1.1.1 or
        (Nk is case 1.1.2 or 1.1.3 and Nk-2 = rotate180(Nk)) or
        (Nk is case 1.3.2 and Nk-2 = rotate90(Nk)) then
            finish  $\leftarrow$  True;
        else if ok = True then
            move according to Nk;
        end
        ok  $\leftarrow$  True;
    else
        if (Nk-1 is case 1.2.4 or 1.3.5) and ((i, j + 1), (i - 1, j), (i - 1, j + 1) are all empty) then
            move to (i - 1, j);
        end
    end
    k  $\leftarrow$  k + 1;
end
```

---

## 1.8 Correctness

We define the *bounding box*  $BB(t)$  of the robots as the smallest enclosing rectangle (oriented with the grid's axes) which contains all robots at step  $t$ .

**Lemma 1.1.** *When following the algorithm described above, the bounding box of the robots is monotonically non-inflating, i.e.,  $BB(t+1) \subseteq BB(t)$  for all  $t$ .*

**Lemma 1.2.** *If there is exactly one robot in the topmost row of the bounding box, then it moves down after at most a constant number of steps.*

## 2 Calcul du temps de rassemblement dans le cas d'un bloc (draft)

### 2.1 Cas paire

Bloc de taille  $n$  par  $n$  avec  $n \geq 3$ .

On ne considère que un côté (les trois autres sont similaires)

$R$  : le plus petit rectangle englobant tous les robots

**Première étape** De bloc à disque

Les deux cases prises par les robots sur les bords du côtés se libèrent à chaque ronde. Donc à chaque ronde, le nombre de robots sur le côté diminue de 2. Le temps pour qu'il ne reste plus que 4 robots est

$$\frac{n-4}{2}$$

**Deuxième étape** Effondrement du disque (ou du cercle)

Le côté ne contient plus que 4 robots adjacents. Donc au bout de deux étapes, il s'effondre. On se rend compte que tant que  $n \geq 3$ , après chaque effondrement, le côté contient de nouveau 4 robots adjacents. Donc, pour que l'espace devienne un bloc de taille 2 par 2 (cas terminal), il faut que chaque côté "descendant" de  $n/2 - 1$ , soit un nombre d'étapes de

$$(\frac{n}{2} - 1) * 2$$

Puis il faut une dernière ronde pour que les robots se rendent compte qu'ils ont terminés.

**Totaux** Le temps total  $T$  pour que un bloc se rassemble est donc :

$$T = (\frac{n}{2} - 1) * 2 + \frac{n-4}{2} + 1$$

$$T = n + \frac{n}{2} - 3$$

**2.2 Cas impaire**

Le cas impaire est assez similaire : On utilise des côtés qui ont 3 robots adjacents, et "descendent" en deux étapes.

**Première étape**

$$\frac{n-3}{2}$$

**Deuxième étape**

$$(\frac{n-1}{2}) * 2$$

Pas de ronde en plus après la deuxième étape.

**2.3 Cas général (paire et impaire)**

$$n + \left\lceil \frac{n}{2} \right\rceil - 3$$

Formule vérifiée pour des blocs de côtés de taille 3 à 50 de côtés de taille 3 à 100.

**2.4 Généralisation aux rectangles**

On considère des rectangles de taille  $m$  par  $n$  avec  $m \leq n$ .

$$\left\lceil \frac{n-4}{2} \right\rceil + 2 * \left\lfloor \frac{m}{2} \right\rfloor - 2 \text{ si } m \text{ pair} + 1 \text{ si } m \text{ ou } n \text{ pair}$$

**2.5 Généralisation aux carrés (blocs vides)**

Tout ce qui a été dit sur les blocs peut s'appliquer (et a été vérifié) aux carrés et aux rectangles. On peut associer les robots qui forment le contour du bloc aux robots qui forment le cercle. Puis, on se rend compte que à chaque étape, les robots correspondant deux à deux prendront la même décision.