



---

## Robotique en essaim

---

Julien ANDRES  
Thomas TAYLOR

Sous la direction de  
Nicolas BREDECHE



25 mai 2018  
Projet ANDROIDE

# Sommaire

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Robotique en essaim . . . . .	2
1.2	Couverture . . . . .	2
1.3	Agrégation . . . . .	2
1.4	Algorithmes d'embodied evolution . . . . .	2
1.5	Problématique . . . . .	2
<b>2</b>	<b>Matériel utilisé</b>	<b>3</b>
2.1	Kilobot . . . . .	3
2.2	Bloc Actif . . . . .	3
2.3	Arène . . . . .	4
2.4	Simulateur . . . . .	4
<b>3</b>	<b>Couverture</b>	<b>5</b>
3.1	Algorithme . . . . .	5
3.2	Résultats . . . . .	6
3.2.1	Commencement groupé . . . . .	6
3.2.2	Commencement éparpillé . . . . .	7
3.2.3	Comparaison . . . . .	8
<b>4</b>	<b>Agrégation</b>	<b>9</b>
4.1	Agrégation naïve . . . . .	9
4.2	Agrégation probabiliste . . . . .	10
4.3	Résultats . . . . .	11
4.3.1	Agrégation naïve . . . . .	11
4.3.2	Agrégation probabiliste - Analyse Probabilités . . . . .	12
4.3.3	Agrégation probabiliste - Analyse Paramètres . . . . .	14
<b>5</b>	<b>mEDEA</b>	<b>15</b>
5.1	Algorithme . . . . .	15
5.2	Résultats . . . . .	17
5.2.1	Nombre de morts . . . . .	17
5.2.2	Survie d'un génome . . . . .	17
5.2.3	Comparaison avec expérience témoin . . . . .	19
<b>6</b>	<b>Conclusion</b>	<b>20</b>
<b>7</b>	<b>Bibliographie</b>	<b>21</b>

# 1 Introduction

## 1.1 Robotique en essaim

La robotique en essaim s'inspire d'observations d'insectes et d'animaux sociaux pour l'étude de comportements collectifs issus de simples tâches individuelles. Les principales propriétés de ce domaine sont la robustesse, grâce à un code décentralisé, et l'échelle proposée, le code pouvant être déployé sur différentes tailles d'essaim.

## 1.2 Couverture

La couverture est un problème fondamental dans la robotique en essaim. En effet, un algorithme de couverture permet aux robots de se déployer dans un environnement inconnu ou dynamique, tout en maintenant un contact entre eux. Ce problème se retrouve dans beaucoup de domaines, comme le sauvetage de personnes en milieu hostile ou inconnu.

## 1.3 Agrégation

L'agrégation est un comportement de base des essaims. Il permet aux organismes de faire face ensemble à un problème. Dans le domaine de la robotique en essaim, ce comportement permet à des robots dispersés dans un environnement de se regrouper en un ou plusieurs groupes.

## 1.4 Algorithmes d'*embodied evolution*

L'objectif de ces algorithmes est d'arriver, à partir d'un génome aléatoire définissant le comportement d'un robot, à un comportement permettant à ce robot de survivre dans son environnement. Chaque Kilobot communique son génome à ses voisins et adopte un des génomes qu'il a récupérés. Sur le long terme, cette sélection permet aux robots de s'adapter à leur environnement.

## 1.5 Problématique

L'objectif de ce projet est de tester le dispositif expérimental de l'ISIR en implémentant sur des Kilobots des comportements simples d'agrégation et de couverture ainsi qu'un comportement plus évolué d'agrégation.

Dans un second temps, nous étudierons un algorithme dit d'*embodied evolution* dirigé par la pression environnemental. Ce type d'algorithme distribué est plus léger et permet de converger vers des comportements opérant un compromis entre survie (proximité d'un point de recharge) et diffusion du jeu de paramètres (permettant à l'essaim une plus grande chance de survie collective).

## 2 Matériel utilisé

### 2.1 Kilobot

Un Kilobot [1] est un robot miniature, créé par l'université d'Harvard. Ces robots permettent l'étude de la robotique en essaim, leur faible coût étant un avantage pour la manipulation d'un grand nombre d'individus.

**Communication** Les robots peuvent communiquer entre eux grâce à des messages qu'ils envoient à leurs voisins à une distance d'environ 7cm. L'intensité du signal permet à un Kilobot d'évaluer la distance à laquelle se trouve la source du message. Chaque robot possède aussi un capteur de luminosité.

**Mouvements** Chaque robot possède deux moteurs, placés de chaque côté de son corps. Ils permettent au robot d'avancer en faisant vibrer ses pattes. Les déplacements d'un Kilobot se font donc en modulant la vitesse de chacun de ses moteurs selon la direction voulue.

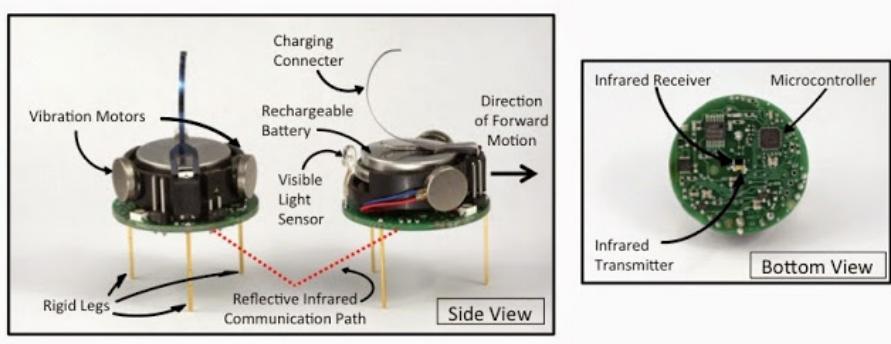


Figure 1: Kilobot

Les comportements des robots sont écrits en C, en utilisant l'API kilolib et peuvent être transmis à l'ensemble des Kilobots grâce à un contrôleur infrarouge.

### 2.2 Bloc Actif

Le bloc actif a été développé par l'ISIR dans le cadre de la recherche sur Kilobots. Il permet d'interagir avec les robots en émettant un signal infrarouge. Le numéro du message et l'intensité du signal peuvent être modulés grâce à des boutons sur le bloc actif. Ce bloc actif fera office d'obstacle et de point de ralliement pour certains de nos algorithmes.



Figure 2: Distance max à intensité max

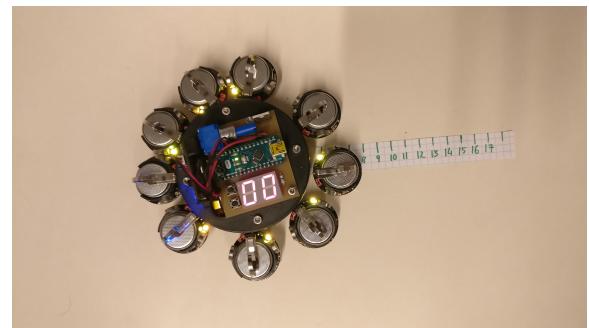


Figure 3: Distance max à intensité min

Les robots devant être vraiment trop collés au bloc pour recevoir le signal en intensité minimale, nous utiliserons dans toutes les expériences l'intensité maximale du bloc actif.

## 2.3 Arène

Les algorithmes ont été testés sur un essaim de 70 robots, fournis par le laboratoire de recherche ISIR. L'arène utilisée pour l'étude des algorithmes faisait 1x1.5 mètres.

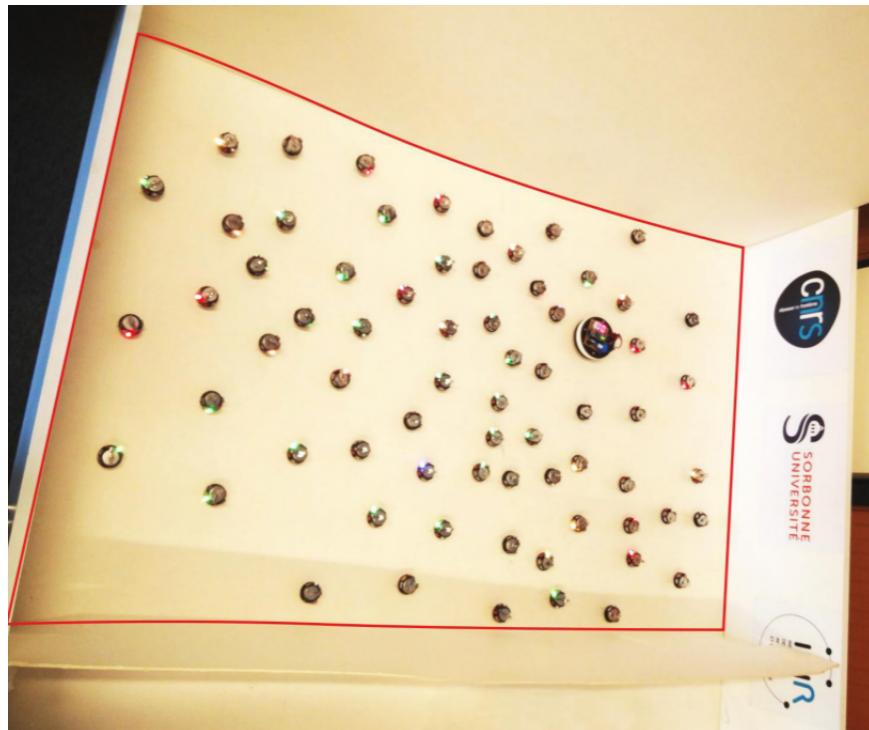


Figure 4: Arène

## 2.4 Simulateur

Afin de faciliter le développement et l'étude des algorithmes, le simulateur Kilombo [2] a été utilisé. Il inclut l'API kilolib et permet ainsi une compatibilité directe du code entre le simulateur et l'essaim de Kilobots.

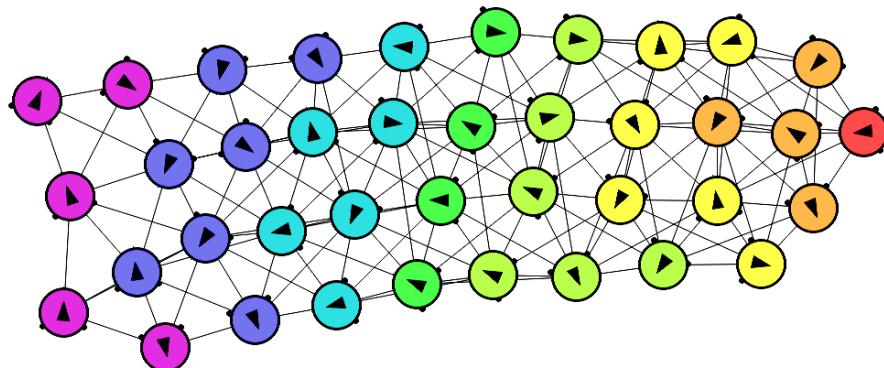


Figure 5: Modélisation d'un gradient sur simulateur

### 3 Couverture

#### 3.1 Algorithme

Cet algorithme est utilisé afin de couvrir la plus grande surface possible, tout en gardant un lien de communication entre les Kilobots. Il est composé de 3 comportements basiques : le déplacement aléatoire(*searching*), l'éloignement (*repel*), et l'attente(*wait*)

**Le déplacement aléatoire** Ce comportement s'effectue lorsque le Kilobot ne détecte aucun voisin. Il bouge alors aléatoirement dans son environnement jusqu'à la détection d'un autre robot. Il passe alors en mode d'attente ou d'éloignement selon la distance à laquelle il est de ses voisins.

**L'éloignement** Si le Kilobot se trouve trop proche de ses voisins, il essayera de s'en éloigner. Soit en s'éloignant de son voisin le plus proche, soit en bougeant aléatoirement jusqu'à être à la distance désirée de tous ses voisins.

**L'attente** Le Kilobot entre en attente quand il se trouve à une distance de tous ses voisins supérieure ou égale à la distance demandée . Il sort de ce comportement s'il ne capte plus aucun voisin ou si un Kilobot se rapproche trop.

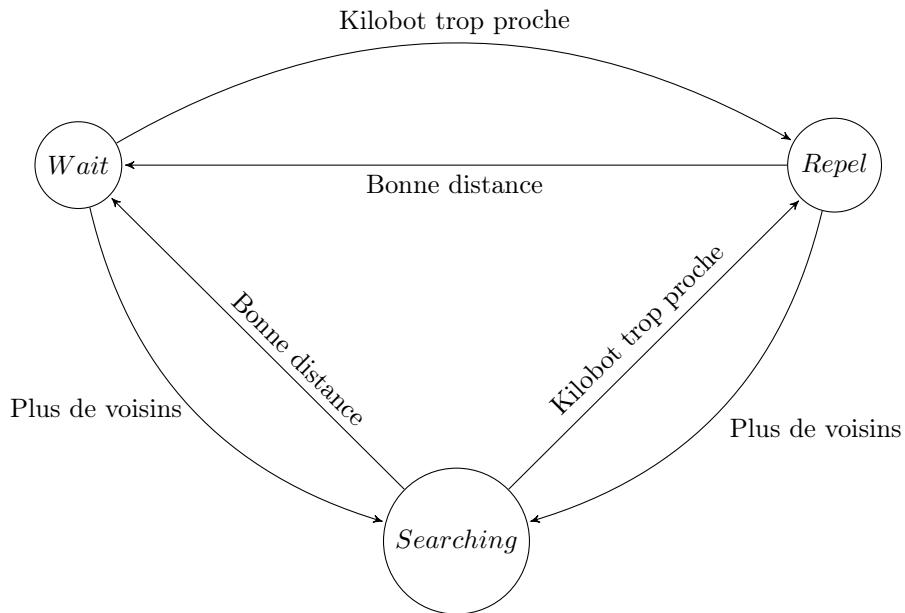


Figure 6: Couverture

## 3.2 Résultats

Deux positions de départ sont possibles pour l'essaim de Kilobots. Nous pouvons partir d'un essaim regroupé en un groupe compact ou d'un essaim éparpillé sur l'ensemble de l'environnement. Nous étudions ici le comportement de l'essaim en fonction de la distance minimale demandée entre chaque robot.



Figure 7: Commencement groupé

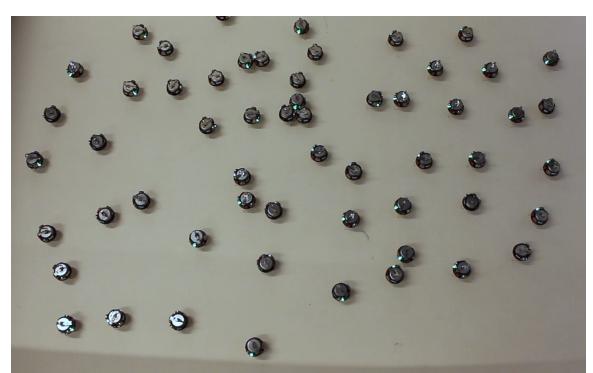


Figure 8: Commencement éparpillé

### 3.2.1 Commencement groupé

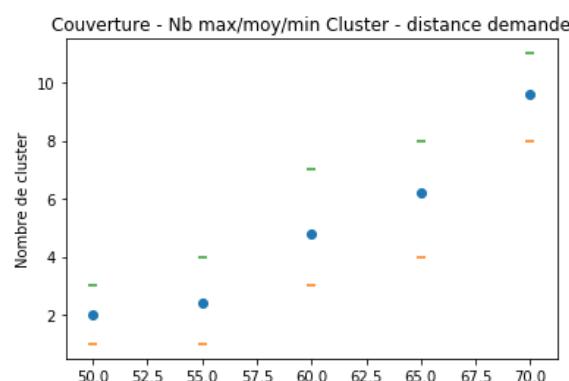


Figure 9: Nombre de clusters selon la distance

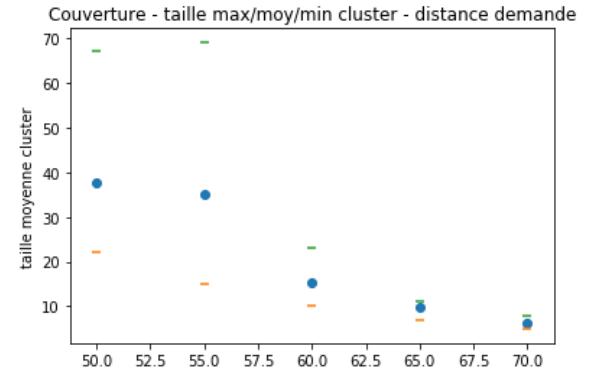


Figure 10: Taille moyenne d'un cluster selon la distance

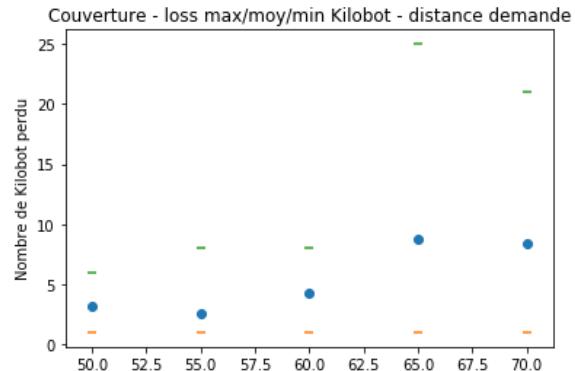


Figure 11: Nombre de Kilobots perdus selon la distance

On remarque ici un nombre très bas de robots *perdus* (c'est-à-dire qui n'appartiennent à aucun groupe).

Le nombre de robots dans un agrégat (c'est à dire un groupe de Kilobots en attente, communiquant tous ensemble) est assez grand pour des valeurs inférieures à 55mm, ce qui s'explique par la proximité des robots dès le départ. Les Kilobots étant proches, il est normal qu'ils soient bien connectés entre eux.

### 3.2.2 Commencement éparpillé

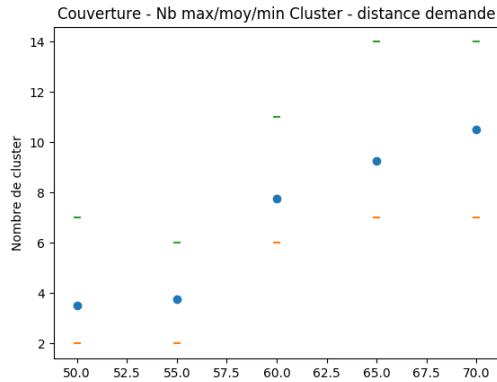


Figure 12: Nombre de clusters selon la distance demandée

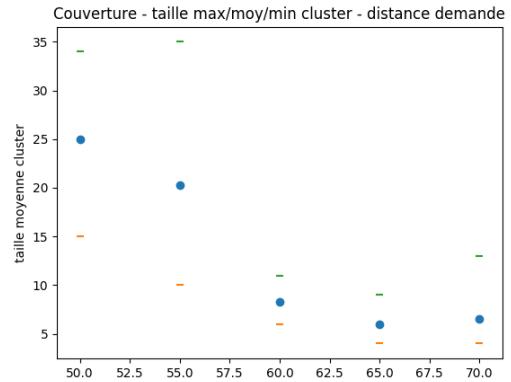


Figure 13: Taille moyenne d'un cluster selon la distance

On remarque immédiatement que plus la distance entre deux robots est grande (elle ne dépasse jamais la distance maximale de communication), moins la couverture est efficace. On remarque que plus la distance demandée est grande, plus le nombre total de groupes est grand et plus leur taille est petite. En effet, le groupe de robots, cherchant à atteindre cette distance minimale, va se fractionner et de nombreux petits groupes se formeront.

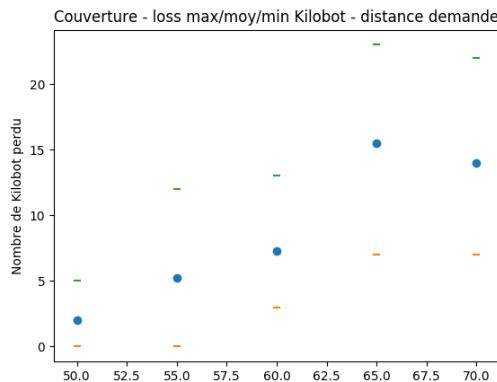


Figure 14: Nombre de Kilobots perdu selon la distance

On remarque aussi que plus la distance est grande, plus le nombre de Kilobots *perdus* est grand. On observe un grand changement entre une distance de 60 et de 65mm. Ce grand écart est aussi observable sur les graphiques précédents.

### **3.2.3 Comparaison**

On remarque que le départ depuis un essaim groupé amène à une configuration des Kilobots plus connectée contenant moins de robots perdus qu'avec un départ éparpillé. En effet, la proximité des Kilobots dès le début de l'expérience les empêches de rester seul.

Cependant, cette connexité se fait au détriment de la durée de convergence de l'algorithme, qui est plus rapide depuis un essaim éparpillé (l'aleatoire ayant déjà fait une partie du travail).

## 4 Agrégation

### 4.1 Agrégation naïve

Le but de cet algorithme est de faire s'agréger les Kilobots vers un objet pouvant représenter une source de nourriture par exemple. Cet objet sera représenté ici par le bloc actif. Le principal but de cet algorithme étant de former un groupe compact de Kilobots autour du bloc actif.

Dans cet algorithme, les robots ne communiquent que pour indiquer l'emplacement du bloc actif. Ils n'ont donc aucune autre information sur leur environnement. Cet algorithme est composé de 3 comportements : la recherche, l'approche et l'émission.

**Recherche** Le robot bouge aléatoirement dans son environnement afin de trouver une source d'émission. Ici, le Kilobot ne fait aucune hypothèse sur son environnement. Il n'émet rien et ne reçoit donc aucune information provenant des autres robots dans le même état que lui. Ce comportement cesse lorsqu'il reçoit un message, le prévenant de la présence du bloc actif ou d'un robot proche de ce bloc actif. Il passe alors en comportement d'approche.

**L'approche** Lors de cette phase, le robot va essayer de s'approcher du signal le plus proche. La méthode d'approche utilisée (qui est la même dans tous les algorithmes qui suivent) consiste à approcher en faisant des rotations. A la réception du message, le robot choisit aléatoirement une direction entre droite et gauche. A chaque nouveau message, si le Kilobot s'est rapproché de la source, il continue dans la même direction, sinon il prend la direction opposée.

Il sort de ce comportement s'il perd le signal de la source (retourne à la recherche) ou s'il est assez proche de tous ses voisins émetteurs. Il passe alors dans l'état d'émission.

**Émission** Le robot se trouve à une distance inférieure ou égale à celle demandée par l'algorithme. Il se met donc à émettre à son tour afin de transmettre l'information aux robots qui lui sont proches.

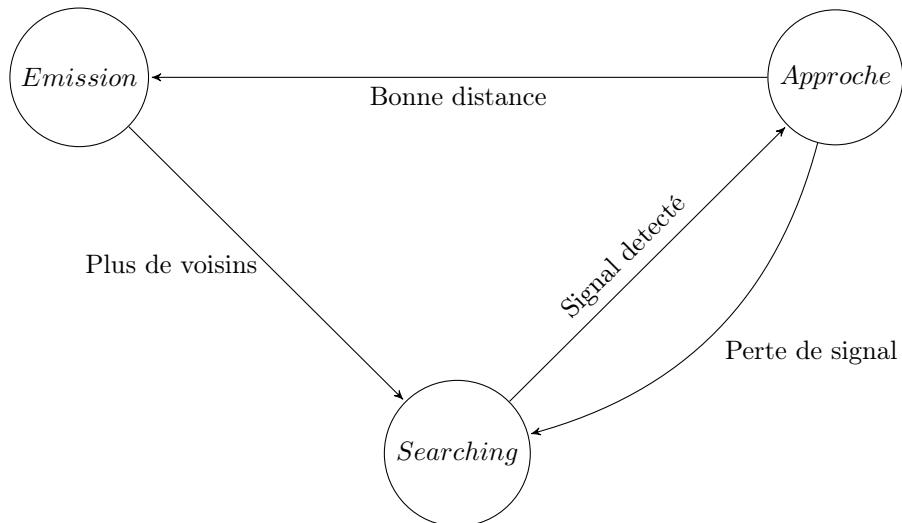


Figure 15: Agrégation naïve

## 4.2 Agrégation probabiliste

Cet algorithme découle de celui présenté par O.Soysal et E.Sahin. [3] Il permet aux robots de s'agglomérer en se basant sur une probabilité calculée indépendamment par chaque Kilobot.

Il est composé de 4 comportements basiques : l'approche (*approach*), l'attente (*wait*), l'éloignement (*repel*) et un évitemenent d'obstacle. Un Kilobot n'ayant pas de capteur de proximité, il lui est impossible de détecter un obstacle tel qu'un mur. L'évitement d'obstacle a donc été remplacé par un déplacement aléatoire dans l'environnement (*searching*).

**Le déplacement aléatoire** Quand le robot est seul dans l'environnement (aucun voisin), il bouge de façon aléatoire. Toutes les secondes, il choisit une direction parmi les 3 disponibles (gauche, droite, tout droit) et la garde jusqu'à la l'actualisation suivante. Dès qu'il détecte un voisin, il passe en comportement d'approche.

**L'approche** Dans ce comportement, le robot décide de s'approcher du voisin le plus intéressant. Le voisin le plus intéressant est défini comme celui ayant le plus de voisins à côté de lui. Si le Kilobot arrive à rejoindre le groupe, ce qui n'est pas obligatoire, à cause de l'absence de capteurs de proximité, il s'arrête et passe dans l'état d'attente.

**L'attente** Quand le robot est dans cet état, il fait partie d'un groupe composé d'au moins 2 individus. Il a cependant une probabilité de quitter ce groupe et d'entrer dans l'état d'éloignement à chaque seconde.

**L'éloignement** Ce comportement fait s'éloigner le robot du groupe auquel il était agrégé précédemment. Pendant toute la durée de cet éloignement, à chaque seconde, le Kilobot a une probabilité de revenir en mode d'approche et de rejoindre le groupe qu'il vient de quitter. Ce comportement s'arrête quand le robot est seul ou que 10 secondes se sont écoulées. Il revient alors en déplacement aléatoire.

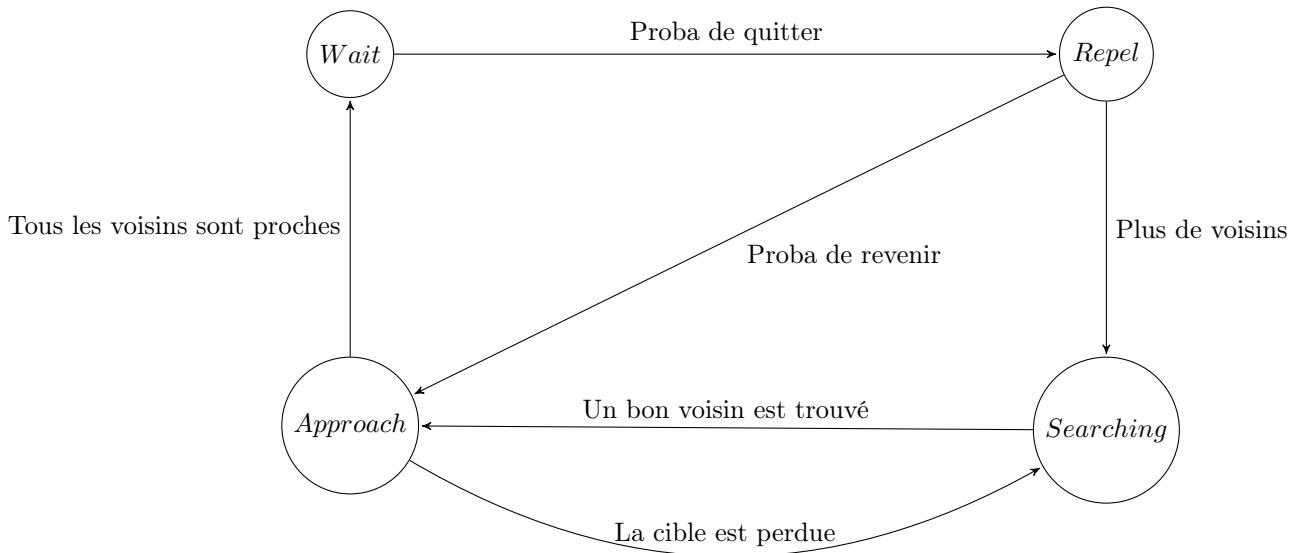


Figure 16: Agrégation probabiliste

## 4.3 Résultats

### 4.3.1 Agrégation naïve

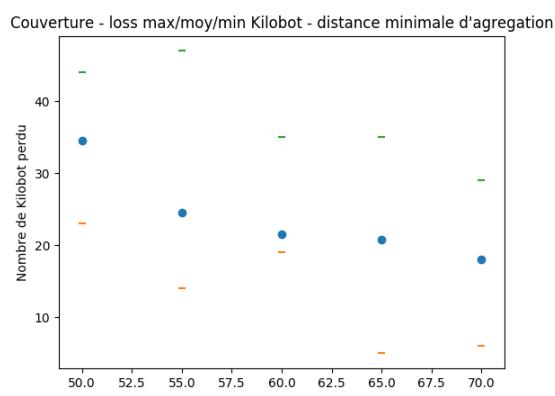


Figure 17: Nombre de robots perdus selon la distance maximum d'agrégation



Figure 18: Photo fin d'expérience d'agrégation naïve

On remarque ici que plus la distance maximale d'agrégation est grande, moins il y a de Kilobots perdus. En effet, ces derniers ayant une distance plus grande pour entrer en attente, ils ont moins de temps d'approche, et donc risquent moins de se perdre lors de cette approche.

Lors des expériences, on remarque que les Kilobots se regroupent en chaines et non en agglomérat. De plus, on remarque que les robots perdus sur les côtés de l'arène ont peu de chance de se sortir de cette bordure. En effet, les robots adjacents les poussent et les empêchent de se libérer.

### 4.3.2 Agrégation probabiliste - Analyse Probabilités

Cet algorithme se base sur la probabilité qu'a un robot de quitter son groupe afin de trouver potentiellement un nouveau groupe composé de plus d'individus (et donc plus intéressant). Le paramètre important de cet algorithme est donc le calcul de cette probabilité. Nous avons ici testé trois façons différentes de calculer cette probabilité.

**Probabilité constante** Nous utilisons ici la probabilité constante proposée par O.Soyal et E.Sahin comme étant celle pour laquelle ils ont obtenu les meilleurs résultats.

$$P(x) = 0.0001$$

**Probabilité suivant taille groupe** Afin de permettre aux gros agglomérats de robots de rester ensemble, nous avons défini la probabilité pour un Kilobot quitte un groupe en fonction du nombre de voisins de celui-ci et par rapport à une constante définie comme la taille d'un groupe que nous considérons *stable*. Or après expérimentation, nous avons remarqué qu'un robot arrivant au bord d'un agglomérat n'y reste pas forcément car il n'est connecté qu'à quelques individus de ce groupe. Afin de traiter les deux cas d'appartenance à un groupe (au centre ou sur un bord), nous prenons donc le maximum entre le nombre de voisin du Kilobot et le nombre maximum de voisin du meilleur de ses voisins. Le meilleur de ses voisins étant défini comme le voisin ayant le plus de voisins.

$$P(x) = \min(1, \frac{v(x)}{\text{TAILLE\_GRAND\_GROUPE}})$$

avec

$$v(x) = \max(\text{nombre\_de\_voisin}(x), \text{nombre\_de\_voisin}(\text{meilleur\_voisin}(x)))$$

**Probabilité mixte** Cette probabilité a été trouvée après avoir vu les résultats expérimentaux des deux probabilités précédentes. Elle a comme objectif de créer des agglomérats stables en ne permettant pas à un robot de partir s'il appartient à un suffisamment grand groupe, et en utilisant la probabilité trouvée par O.Soyal et E.Sahin. Elle est définie comme :

$$P(x) = \begin{cases} 0 & \text{si } v(x) \geq \text{TAILLE\_GRAND\_GROUPE} \\ 0.0001 & \text{sinon} \end{cases}$$

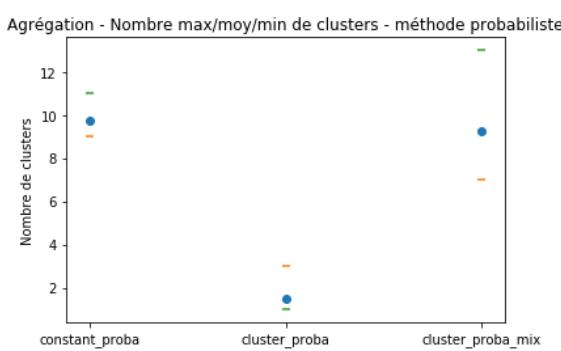


Figure 19: Nombre de clusters selon la probabilité

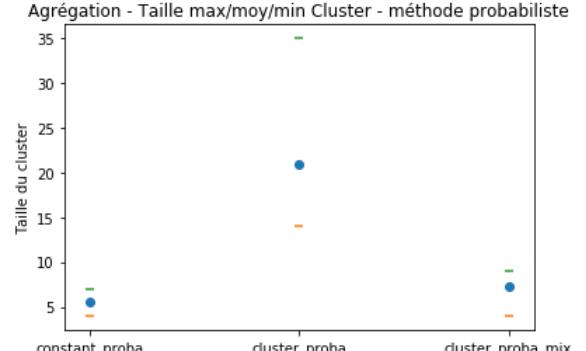


Figure 20: Taille moyenne d'un cluster selon la probabilité

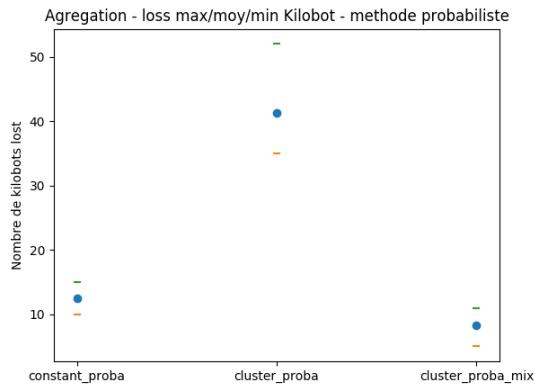


Figure 21: Nombre de robots perdus selon la probabilité



Figure 22: Photo de fin expérience d'agrégation linéaire

On observe que la probabilité constante, proposée par O.Soysal et E.Sahin, donne un grand nombre de petits groupes. En effet, les agrégats sont formés rapidement et la faible probabilité qu'a un robot de partir ne permet pas la mise en place d'un grand groupe.  
En revanche, la probabilité suivant la taille du groupe permet de remédier à ce problème. On observe en effet un petit nombre de groupes composés d'un grand nombre d'individu. Un individu dans un petit groupe a en effet une forte probabilité d'en sortir. Cependant, cette mesure de probabilité ne domine pas la première, le nombre de robots perdus étant largement supérieur à celle ci.

### 4.3.3 Agrégation probabiliste - Analyse Paramètres

Afin de remédier aux défauts de la probabilité linéaire suivant la taille d'un grand agrégat, nous avons essayé d'étudier les effets de cette taille TAILLE\_GRAND\_GROUPE, afin de faciliter la création d'un groupe robuste et de minimiser le nombre de robots perdus.

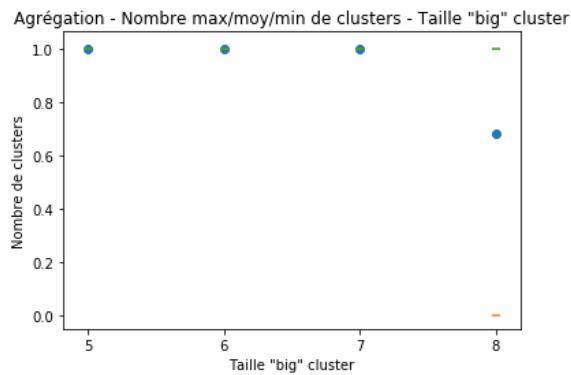


Figure 23: Nombre de clusters selon TAILLE\_GRAND\_GROUPE

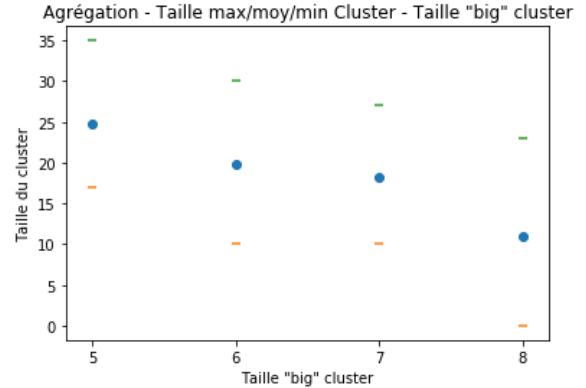


Figure 24: Taille moyenne d'un cluster selon TAILLE\_GRAND\_GROUPE

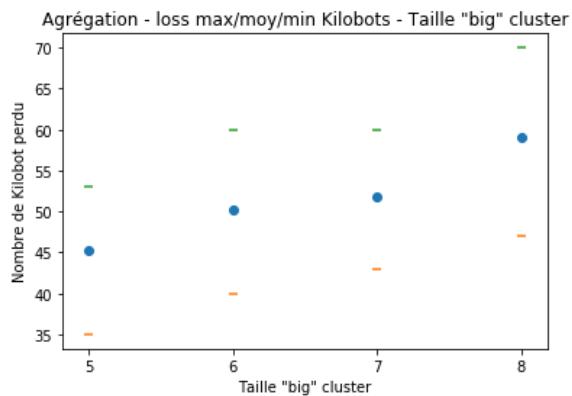


Figure 25: Nombre de robots perdus selon TAILLE\_GRAND\_GROUPE

On remarque ici que plus la valeur de TAILLE\_GRAND\_GROUPE est faible, moins les robots se perdent. Cependant, une petite valeur ne donne pas forcément plus de groupes.

Une valeur trop haute, en revanche, induit un grand nombre de Kilobots perdus et une taille de groupe plus petite. En effet, un Kilobot a rarement plus de 6 ou 7 voisins autour de lui. La probabilité ne permet donc pas la création d'agrégats stables.

## 5 mEDEA

Nous implémentons ici la version Vanilla-EE de l'algorithme mEDEA présenté par J.Montanier, S.Carrignon et N.Bredeche [4]. Il permet à un essaim de robots de s'adapter à leur environnement en fonction des contraintes de celui-ci.

### 5.1 Algorithme

```

1      Initialisation aléatoire du génome;
2      while True do
3          for i=0 to lifetime do
4              if self.genome!=NULL then
5                  move() //selon le génome
6                  fitness=calculerFitness()
7                  transmettre(génome,fitness) //aux voisins accessibles
8
9              end if
10             genomeQueue=receptionGénome() //traitement des messages
11             if genomeQueue!=NULL then
12                 genomeList.add(genomeQueue)
13             end if
14         end for
15         self.genome=NULL
16         if genomeList.size > 0 then
17             newGenome=select(genomeList) //selection selon meilleure fitness
18             newGenome=mutation(newGenome) //proba de muter
19             self.genome=newGenome
20         end if
21         genomeList.empty()
22     end while
23

```

Figure 26: Vanilla-EE

Dans notre cas, l'algorithme est exécuté sur chaque robot et évalue le comportement du robot pendant ses déplacements. Un robot correspond donc aux paramètres du comportement courant. Un Kilobot ne dispose que de 3 capteurs d'entrée :

- La luminosité
- Les messages reçus
- La distance du message à l'envoyeur(calculée par le Kilobot)

Il nous a donc fallu créer des capteurs *artificiels* via le transfert d'information entre les Kilobots par message. Notre génome est donc constitué des poids d'un perceptron sans couches cachées. Les entrées sont donc :

- Une entrée active en permanence comme neurone de biais
- La moyenne des distances à tous les voisins connus
- Le nombre de voisins actuels
- Une entrée binaire associée à la présence ou non du bloc actif

Il y a deux sorties, correspondants aux moteurs gauche et droit de chaque Kilobot. Cela nous donne donc un génome de 8 gènes correspondant aux 8 paramètres du perceptron. Chaque paramètre peut prendre 3 valeurs : {-1,0,1}.

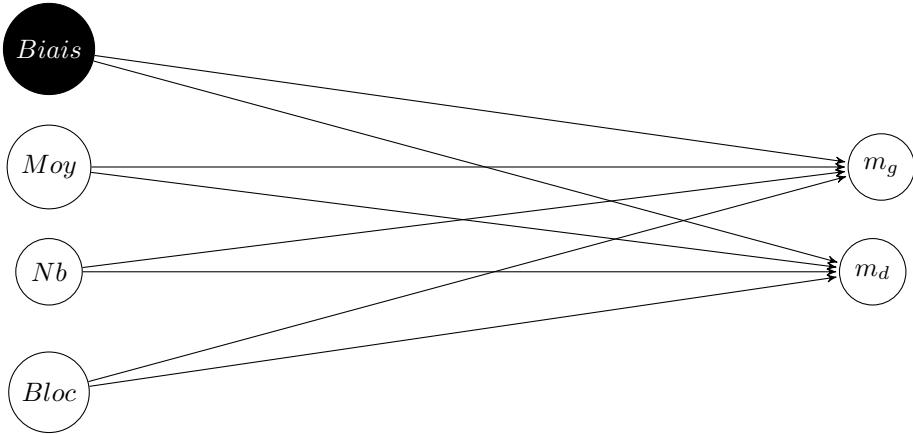


Figure 27: Contrôleur mEDEA

**Fonction d'activation** Notre fonction d'activation permet d'obtenir une sortie dans l'intervalle de vitesse des moteurs :

$$A(x) = \frac{1}{1+e^{-x}} * (Max\_moteur - Min\_moteur) + Min\_moteur$$

### Fitness

$$F(x) = \sum_i f(x) \quad \forall i \in 0, 1, \dots, evalTime$$

avec

$$f(x) = \begin{cases} 1 & \text{si Block actif detecte} \\ 0 & \text{sinon} \end{cases}$$

Notre fitness est ici calculée sur les 40 dernières demi-secondes (fenêtre glissante). Elle est mise à jour à chaque envoi de message du Kilobot vers ses voisins.

**Mutation** La probabilité de mutation est un facteur important de l'algorithme. Une probabilité trop grande peut mener à une population incapable de se stabiliser alors qu'un taux trop bas peut amener à une absence de génome intéressant.

```

1   if random() < P_mutation then
2       for (i=0;i<len(genome);i++) do
3           if random() < 1/len(genome) then
4               modif=randint(1,2)
5               genome[i]=(genome[i]+modif+1) mod 3 -1
6           end if
7       end for
8   end if
9

```

Figure 28: Mutation

Après la décision d'une mutation sur le génome, nous avons décidé de mettre une probabilité fixe de  $\frac{1}{\text{nombredegnes}}$  par gène. Cela permet d'avoir en moyenne un seul gène changé par mutation afin de ne pas avoir un génome changeant drastiquement d'une génération à une autre.

## 5.2 Résultats

### 5.2.1 Nombre de morts

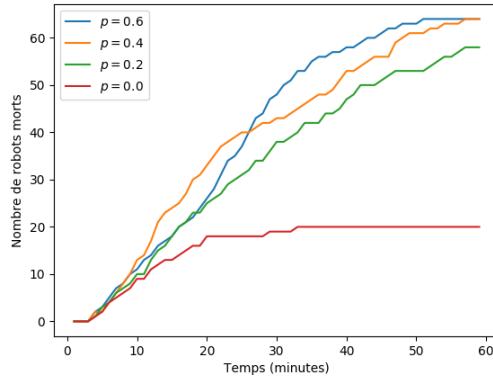


Figure 29: Nombre de robots morts par génération

Ces expériences ont été faites en environnement fini, avec sur une durée d'une heure. On remarque la grande différence entre le nombre de morts avec une probabilité de mutation nulle et non nulle.

En effet, comme les génomes mutent pas, une fois qu'une configuration stable a été trouvée, les robots y restent. On observe en effet pendant l'expérience des agglomérats de robots qui se transfèrent le même génome. Ce comportement est impossible dans le cas d'une probabilité non nulle de mutation, un robot ayant une chance non nulle de partir de cette configuration.

### 5.2.2 Survie d'un génome

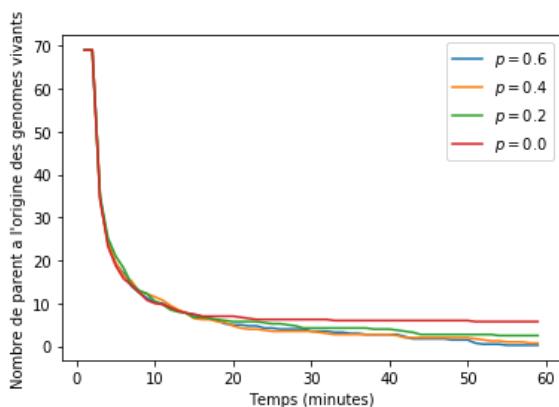


Figure 30: Nombre de parents à l'origine des génomes des robots vivants par génération

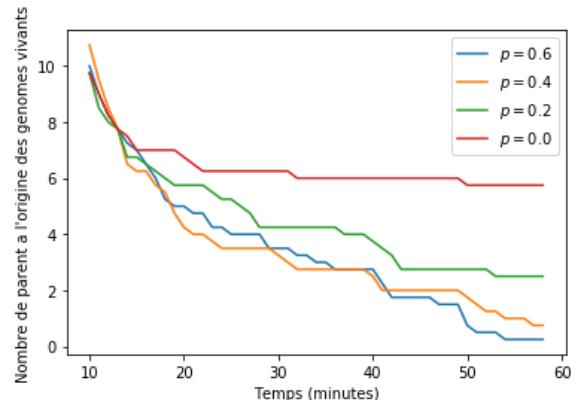


Figure 31: Nombre de parents à l'origine des génomes des robots vivants par génération (zoom)

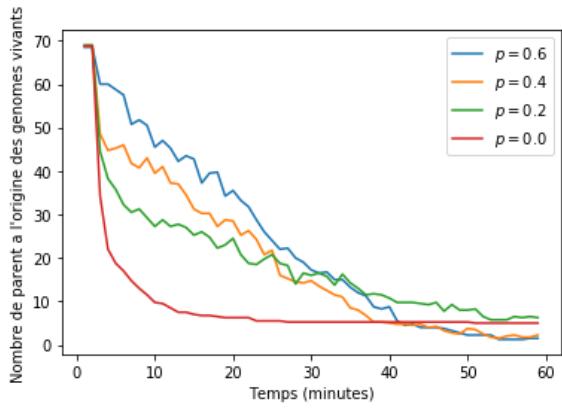


Figure 32: Nombre de génomes différents par génération

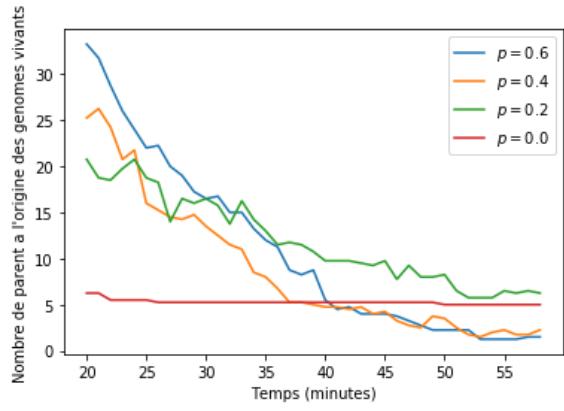


Figure 33: Nombre de génomes différents par génération (zoom)

Le nombre de génomes différents par population dans le cas d'une probabilité de mutation nulle baisse de façon très rapide. C'est à partir de la 20<sup>e</sup> génération que le nombre de génomes reste stable. Couplé avec le graphique sur le nombre de robots morts, on en déduit une stabilisation rapide de la population. Ce qui se traduit visuellement par des agglomérats de robots, tournant sur eux-mêmes et ayant tous le même génome.

Dans le cas d'une probabilité non nulle de mutation, on remarque que les probabilités de 0.4 et 0.6% donnent les mêmes résultats : une population se diversifiant au cours des générations, sans parvenir à rester stable. On peut cependant remarquer qu'à partir de la 20<sup>e</sup> génération, les 40 génomes différents de la population sont issus d'uniquement 5 à 6 parents différents. Cette diversité se fait donc principalement grâce aux mutations.

### 5.2.3 Comparaison avec expérience témoin

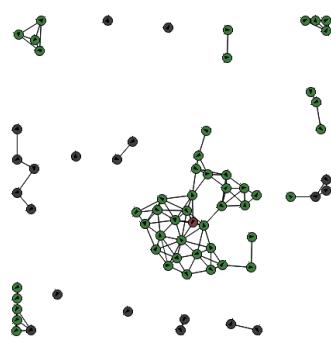


Figure 34: Résultat d'expérience mEDEA avec bloc actif

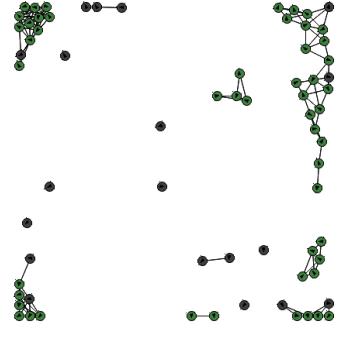


Figure 35: Résultat d'expérience mEDEA témoin sans bloc actif

Grâce aux exécutions répétées de mEDEA, on a pu observer un comportement récurrent des Kilobots près du bloc actif : Si dès le commencement de l'expérience, il n'y a pas de groupe de robot présent près du bloc actif il est très peu probable de voir un agglomérat de Kilobots se former proche de ce bloc. Cependant, si un Kilobot génère un bon génome lui permettant de rester près de ce bloc, alors la dispersion de ce dernier dans la population permettra de voir rapidement se former un groupe près du bloc.

On observe lors des expériences sans bloc actif (ici représenté par un Kilobot rouge ne faisant qu'émettre), que les murs "remplacent" ce bloc actif et permettent une aggrégation efficace des robots.

## 6 Conclusion

L'agrégation et la couverture sont des comportements fondamentaux de la robotique en essaim. Les Kilobots offrent un moyen robuste d'étude de ces comportements par leur simplicité et accessibilité. Ils permettent une implémentation rapide d'un algorithme distribué tel que mEDEA. Ont été étudiés deux types d'agrégation, une couverture ainsi qu'un algorithme d'embodied evolution.

L'analyse de ces comportements sur robots réels illustre l'avantage d'un simulateur. En effet, le calcul de la distance d'un message reçu peut être défaillant (en raison d'obstructions, de la qualité de la surface utilisée ou d'un défaut matériel) et entraîner une perturbation quant aux résultats obtenus. D'autres facteurs sont à prendre en compte lors de la comparaison de résultats obtenus sur simulateur et en réalité : le bloc actif n'est pas modélisable dans le simulateur et a été remplacé par un Kilobot immobile émettant le signal. Il est donc susceptible d'être heurté et déplacé, ce qui n'est pas possible sur le vrai modèle.

Les Kilobots ne possède pas de capteur de proximité, l'évitement d'obstacle est impossible : les robots en viennent à se bloquer entre eux ou contre les murs. Des bandes de LEDs (toujours en phase production à l'ISIR) permettraient aux Kilobots de recevoir des messages des obstacles et d'avoir ainsi une meilleure perception de l'environnement. Dans le cas de mEDEA, elles apporteraient une entrée supplémentaire au perceptron (et par la même, une solution au manque de capteurs constaté sur les Kilobots). Grâce à ce dispositif, l'étude des algorithmes déployés sera plus précise, ceux-ci contenant majoritairement une marche aléatoire destinée à l'exploration. La probabilité de mutation constitue le point central de mEDEA dans l'analyse des résultats. Un plus grand nombre d'expériences est nécessaire pour obtenir une probabilité de mutation faisant la balance entre diversité génétique et stabilité de la population.

Enfin, un Docker (plateforme de conteneurisation) a été développé et utilisé tout au long du projet. Il contient les packages et applications à utiliser lors de la manipulation des Kilobots, du développement d'un programme à son envoi aux robots en passant par le simulateur. Cette plateforme, baptisée kilodocker, a pour but de faciliter la prise en main des robots et de ses outils pour le futur utilisateur.

## 7 Bibliographie

### Références

- [1] M. Rubenstein, C. Ahler, and R. Nagpal, “Kilobot: A low cost scalable robot system for collective behaviors”, in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 3293–3298.
- [2] F. Jansson, M. Hartley, M. Hinsch, I. Slavkov, N. Carranza, T. S. G. Olsson, R. M. Dries, J. H. Grönqvist, A. F. M. Marée, J. Sharpe, J. A. Kaandorp, and V. A. Grieneisen, “Kilombo: A Kilobot simulator to enable effective research in swarm robotics”, Nov. 2015.
- [3] O. Soysal and E. Sahin, “Probabilistic aggregation strategies in swarm robotic systems”, in *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.*, pp. 325–332.
- [4] J.-M. Montanier, S. Carrignon, and N. Bredeche, “Behavioral Specialization in Embodied Evolutionary Robotics: Why So Difficult?”, *Frontiers in Robotics and AI*, vol. 3, 2016.