

Recherche de consensus en robotique en essaim

présenté par LY Jean-Baptiste
M1 ANDROIDE

dans le cadre de l'UE P-ANROIDE encadrée par
BREDECHE Nicolas et MAUDET Nicolas

Juin 2020

Table des matières

Introduction	1
1 Présentation	3
1.1 Le kilobot : un mini robot	3
1.2 Kilombo : un simulateur de kilobots	3
1.3 L'arène réelle	4
1.4 L'arène simulée sur Kilombo	5
1.5 Dissémination et exploration	5
2 Les algorithmes dédiés	6
2.1 Descriptions	6
2.1.1 Majority rule	6
2.1.2 Voter model	6
2.2 Cas du regroupement autour de la ressource de plus grande valeur	6
2.3 Cas du pro-rata	7
2.3.1 Description	7
2.3.2 Les agents "têtu" ou "explorateurs"	8
2.3.3 Algorithme minimal	8
2.3.4 Résultats de l'algorithme minimal	9
2.3.5 Algorithme amélioré	12

Introduction

Le sujet de projet a pour l'objet l'étude du problème du *best-of-n* en robotique en essaim. Le problème du *best-of-n*, consiste à une prise de décision collective au sein d'un ensemble de robots aux capacités de communication et de calcul limitées. Parmi n options disponibles, l'essaim doit choisir l'option qui offre la meilleure solution possible afin de satisfaire leurs besoins actuels.

L'objectif de ce projet est d'étudier l'émergence de consensus au sein de l'essaim, c'est-à-dire une prise de décision collective résultant à un accord commun parmi tous les agents. Le sujet s'inspire du comportement collectif des insectes sociaux tels que les abeilles et les fourmis. Il permet alors de fusionner plusieurs domaines ensemble tels que les systèmes multi-agents, l'éthologie et la biologie. Afin de simuler ces situations, le projet devait initialement se mener sur des robots appelés "Kilobots", mais finalement il a été mené sur un de ses simulateurs, appelé nommé "Kilombo".

Dans un premier temps, il s'agit d'implémenter un algorithme existant permettant d'atteindre de manière distribuée un consensus entre deux ressources. Dans un second temps, il est question d'implémenter un algorithme d'apprentissage afin d'évaluer les difficultés que peuvent poser l'apprentissage de consensus.

L'espace de recherche est défini à partir des comportements de phototaxis (les robots se dirigent vers une source de lumière), anti-phototaxis (les robots fuient la lumière) et déambulation libre (les robots se dirigent de manière aléatoire). Il s'agira d'apprendre les conditions de transitions entre chaque comportement. On s'intéresse d'abord au cas où l'essaim doit se regrouper autour de la ressource de plus grande valeur, puis le cas où l'essaim doit distribuer ses forces au pro-rata de la valeur de chaque ressource.

Ce projet inclut ces trois articles et se basera sur ces derniers :

* Valentini et al. (2016) Collective decision with 100 Kilobots : Speed versus accuracy in binary discrimination problems. AAMAS.

* Valentini et al (2017) The best-of-n problem in robot swarms : Formalization, state of the art, and novel perspectives. Frontiers in AI and Robotics.

* Bredeche et al. (2012) Environment-driven distributed evolutionary adaptation in a population of autonomous robotic agents. MCMDS.

Chapitre 1

Présentation

1.1 Le kilobot : un mini robot

Les kilobots sont des mini robots possédant un diamètre de 33mm, destinés à la robotique en essaim. Ils ont été élaborés par l'Université d'Harvard dans le but de pouvoir appliquer des algorithmes utilisant des dizaines voire des centaines de robots, tout en réquerant un faible coût. Ils contiennent un microcontrôleur Atmel ATmega328P, qui est programmable en langage C. Les robots sont équipés de LED, de capteurs de lumière ambiante et de des installations de communication infrarouge à courte portée. Les kilobots peuvent communiquer avec leurs voisins jusqu'à une distance de 7 cm en réfléchissant la lumière infrarouge (IR) sur la surface du sol. Ils se déplacent sur des tiges métalliques raides à l'aide de deux moteurs à vibration.

FIGURE 1.1 – Un kilobot



1.2 Kilombo : un simulateur de kilobots

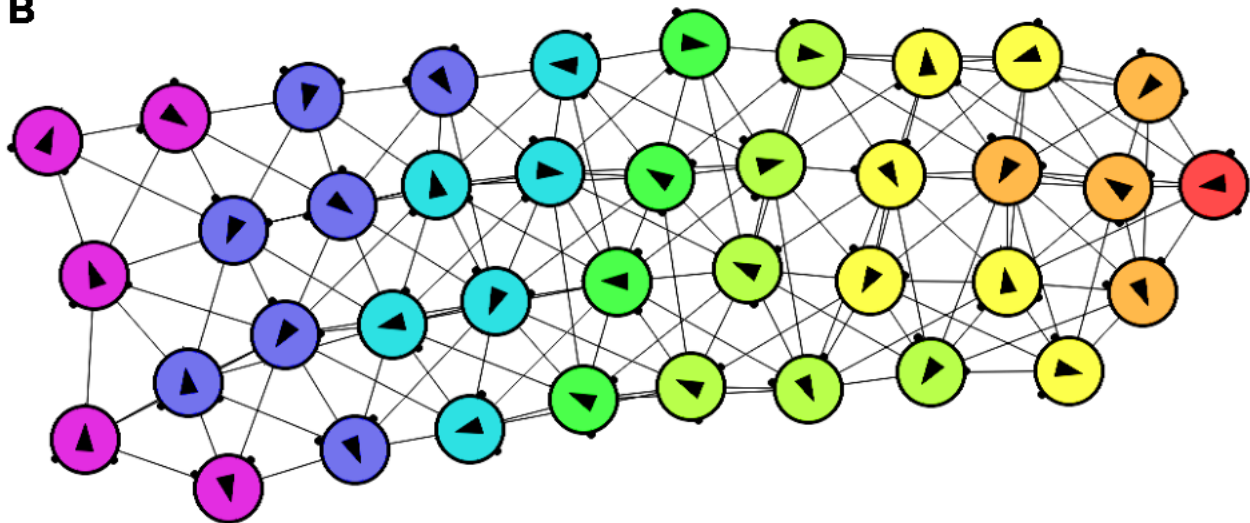
Le simulateur Kilombo permet de simuler efficacement les mouvements de kilobots, et permettent la possibilité d'accélérer leurs mouvements tout en veillant à une bonne simulation. Ainsi cela permet un gain de temps considérable par rapport aux vrais kilobots. De plus, le langage d'implémentation de Kilombo est le même que celui des kilobots, le langage C. Ce qui permet une bonne portabilité pour le code entre les deux plateformes : les vrais kilobots et leur simulateur. Les expérimentations du projet ont été faites exclusivement dessus.

FIGURE 1.2 – Exemple d'un essaim de robots propageant un signal qui forme un gradient. (A) Le programme fonctionnant sur des Kilobots réels ; (B) Le même programme fonctionnant dans le simulateur. [?]

A



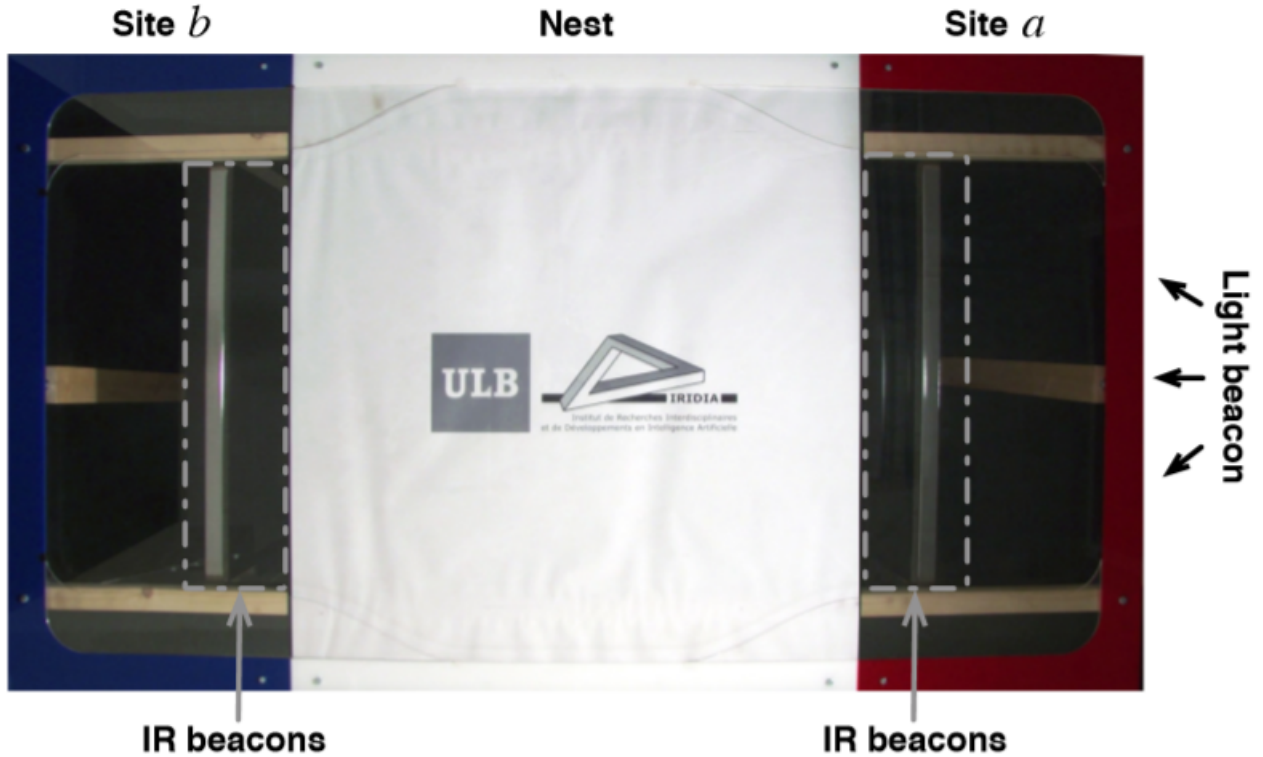
B



1.3 L'arène réelle

Tout au long du projet, les expériences ont été faites dans le cadre d'une reproduction de l'expérience de Gabriele Valentini, Eliseo Ferrante, Heiko Hamann et Marco Dorigo [?]. Il s'agit d'une arène rectangulaire de taille $100 \times 190 \text{ cm}^2$ avec comme surface de déplacements une plaque de plexiglass. L'arène est composée d'une zone centrale appelée "le nid" (appelée aussi zone de négociation ou de dissémination) où est placée en-dessous du plexiglass une surface opaque, et de n sites représentant les ressources à explorer, toutes à égales distances de la zone de négociation. Dans le cas de 2 sites à explorer par exemple, ils sont placés à gauche et à droite de la zone de dissémination, et mesurent chacun $80 \times 45 \text{ cm}^2$. Le nid est la seule zone où les agents pourront prendre leur décision. Les kilobots sont initialement placés dans le nid avec une position et orientation aléatoires. Chaque site possède des lumières infrarouges qui donnent la qualité du site, ces lumières infrarouges sont données par des kilobots retournés.

FIGURE 1.3 – L'arène réelle [?]



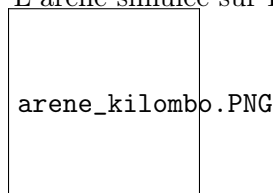
1.4 L'arène simulée sur Kilombo

Dans le cadre du simulateur Kilombo, il est impossible de reproduire exactement les conditions réelles de l'arène. Ainsi il a fallu "tricher" pour parvenir à reproduire le principe de l'expérience réelle. Pour cela, on affecte une lumière différente à toutes les zones de l'arène (nid, sites...). Lorsqu'un kilobot veut rejoindre une telle zone, il sera attiré par phototaxis par la lumière de la dite-zone. Si cette dernière est un site, il connaîtra directement la valeur du site (il n'y aura pas de mesure concrète comme sur la vraie arène).

1.5 Dissémination et exploration

L'expérience est composée de deux phases qui se suivent et recommencent. La première phase nommée "dissémination" ou "négociation" consiste au rassemblement des kilobots au nid. Lors de cette étape, le kilobot retourne au nid pour partager son avis aux autres agents, et pour observer les avis de ses voisins afin de prendre une décision pour la seconde phase qu'est "l'exploration". Cette phase permet au kilobot, en fonction de sa prise de décision lors de la négociation, d'aller explorer explorer le site voulu.

FIGURE 1.4 – L'arène simulée sur Kilombo



Chapitre 2

Les algorithmes dédiés

2.1 Descriptions

2.1.1 Majority rule

La stratégie de prise de décision collective nommée *Majority rule* consiste à prendre la décision majoritaire en fonction des voisins du kilobot rencontrés lors de la négociation dans le nid [?]. Lors de la négociation, l'agent observe l'avis de ses voisins et partage le sien. Par exemple pour un cas où il y a deux ressources a et b à explorer, s'il rencontre plus d'agents pour l'opinion a que pour l'opinion b , alors il ira au site a à la prochaine phase d'exploration, et inversement. En revanche s'il en rencontre autant d'agents pour la ressource a ou b , alors il gardera l'avis qu'il avait à la précédente exploration. Le temps de négociation est proportionnellement aussi élevé que la qualité du site qu'il l'a visitée précédemment. Plus la qualité du site est élevée, plus il restera longtemps en phase de négociation, et meilleure sera sa propagation d'opinion.

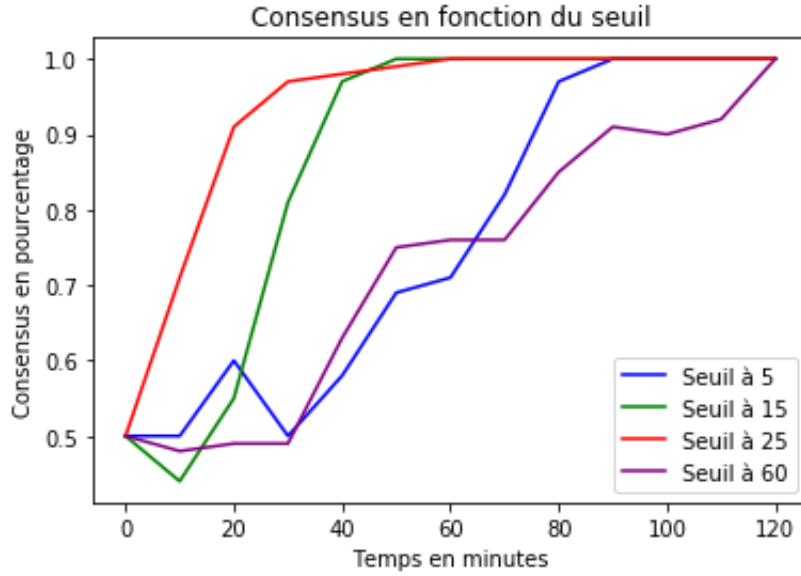
2.1.2 Voter model

La stratégie de prise de décision collective nommée *Voter model* consiste à prendre une opinion aléatoire parmi les agents rencontrés lors de la négociation. Ainsi dans le cas de deux sites a et b par exemple, plus il y a de voisins pour le site a en négociation, plus la probabilité de choisir le site a sera meilleure. Le temps de la phase de négociation se fait identiquement à la stratégie du *Majority rule*.

2.2 Cas du regroupement autour de la ressource de plus grande valeur

Ce cas a déjà été expérimenté par Gabriele Valentini, Eliseo Ferrante, Heiko Hamann et Marco Dorigo [?] [?] avec les stratégies précédemment décrites, sachant que la meilleure est la stratégie du *Majority rule* [?]. L'objectif serait de réappliquer dans ce cas différents seuils décrits dans l'article [?] pour la stratégie du *Majority rule*, dans le cas où les qualités des ressources ne changent pas durant le temps. Les seuils utilisés sont de 5, 15, 25 et 60. Un seuil de k signifierait que dès qu'un agent rencontre k voisins différents, il prendra une nouvelle décision selon la stratégie du *Majority rule*. Si durant le temps de négociation, il n'atteint pas ces k voisins, alors il appliquera la règle à la fin du temps de négociation. Comme on peut constater sur la figure ci-dessous, un seuil de 60, qui équivaut à un seuil inexistant car rarement atteint lors de la négociation, met le plus de temps pour atteindre un consensus. Néanmoins si le seuil est trop faible, par exemple de 5, le consensus prendra du temps à être atteint, dû à la faible précision de prise de décision des agents. Selon l'article [?], le seuil de 25 serait optimal, ce qui montre aussi le graphe ci-dessous.

FIGURE 2.1 – Courbes des seuils du Majority rule



2.3 Cas du pro-rata

2.3.1 Description

Pour l'instant seul le cas classique d'un regroupement général des agents autour de la meilleure ressource a été étudié. Il serait intéressant de s'occuper du cas où l'essaim doit distribuer ses forces au pro-rata de la valeur de chaque ressource, c'est-à-dire la distribution spatiale entre les deux ressources devra être à l'image de la valeur de chacune. Par exemple avec 100 agents, si deux sites a et b ont leur qualité respective 1 et 2, alors cela signifie qu'il y aurait théoriquement 33 agents pour le site a , et 67 agents pour le site b . Soit $theorique_s$ le nombre théorique d'agents qui devrait y être pour atteindre le meilleur consensus au pro-rata du site s , q_s la qualité de la ressource s , qs_{total} la somme de toutes des valeurs de qualité de chacune des ressources disponibles, et $nombre_agents$ le nombre d'agents. D'une manière générale :

$$theorique_s = \frac{q_s}{qs_{total}} \times nombre_agents$$

Ainsi pour la suite avec 100 agents (99 agents pour faciliter l'implémentation avec 3 ressources) voici des tableaux récapitulant le snombre d'agents théoriques en fonction des qualités des ressources.

2 ressources	$theorique_a$	$theorique_b$
$q_a = 1 ; q_b = 2$	33	67
$q_a = 1 ; q_b = 3$	25	75
$q_a = 1 ; q_b = 4$	20	80

3 ressources	$theorique_a$	$theorique_b$	$theorique_c$
$q_a = 1 ; q_b = 2 ; q_c = 3$	17	33	50
$q_a = 1 ; q_b = 3 ; q_c = 6$	10	30	60

4 ressources	$theorique_a$	$theorique_b$	$theorique_c$	$theorique_d$
$q_a = 1 ; q_b = 2 ; q_c = 3 ; q_d = 4$	10	20	30	40
$q_a = 1 ; q_b = 3 ; q_c = 5 ; q_d = 7$	6	19	31	44

Chaque site s possède individuellement une qualité de consensus qui se calcule ainsi : C'est-à-dire que si le nombre d'agents théoriques pour tel site est égal au nombre d'agents expérimental, alors la qualité du consensus de ce site est maximale, égale à 1. Soit $qualite_consensus_s$ la qualité du consensus du site s , $theorique_s$ le nombre théorique d'agents qui devrait y être pour atteindre le meilleur consensus au pro-rata du site s , $experimental_s$ le nombre réel ou expérimental d'agents ayant l'opinion s . Si $experimental_s/2$ est supérieur ou égal à $theorique_s$, alors $qualite_consensus_s$

= 0. Sinon :

$$qualite_consensus_s = \frac{theorique_s - |theorique_s - experimental_s|}{theorique_s}$$

2.3.2 Les agents "têtus" ou "explorateurs"

Afin de parvenir à un consensus au pro-rata, il faut une certaine flexibilité parmi les opinions des agents. Eliseo Ferrante a étudié l'aspect des sites possédant des qualités dynamiques [?], c'est-à-dire des qualités qui peuvent changer au cours du temps.

Pour cela, il a mis en pratique la notion des agents têtus. Ces derniers sont des agents qui n'ont pas la capacité de changer d'avis, ils gardent leur avis mais peuvent néanmoins le partager aux autres agents lors de la négociation. Dans le cas des sites dynamiques [?], l'objectif était d'établir un consensus alors que la qualité des sites variait au cours du temps. Si on appliquait nos différentes stratégies telles quelles dans ce genre de situation, un blocage du consensus pourrait avoir lieu lors du changement de qualité des ressources. En effet, si le consensus arrive très tôt, cela signifierait alors que tous les agents auraient le même avis, et donc ne pourraient plus jamais changer d'avis et seraient bloquer sur leur ressource alors que la qualité de cette dernière aurait diminué. Il faudrait donner la possibilité aux agents de changer de consensus même s'il était considéré comme définitif. Pour remédier à ce problème, on utilise alors les agents têtus combinés à la stratégie du *voter model*. On initialise un agent têtu pur chaque ressource, et ainsi lors du changement de qualité des ressources, l'agent têtu étant pour la ressource ayant la meilleure qualité, restera ainsi plus longtemps dans le nid lors de la dissémination afin de davantage partager son opinion à ses voisins. Ainsi ces derniers auront une certaine probabilité (grâce au *voter model*, cela ne marcherait pas avec la stratégie du *Majority rule*, au début faible, pour changer d'opinion, jusqu'à un consensus inverse général total.

Cette expérience montre que cette notion d'agents têtus combinée avec le *voter model* nous offre une certaine flexibilité pour le consensus.

2.3.3 Algorithme minimal

L'idée serait d'utiliser le fait que les agents ne peuvent plus changer d'opinion afin de converger vers un consensus au pro-rata. Dans le cas précédent du regroupement autour de la plus grande ressource, il était inutile de se préoccuper de ça car cela se faisait automatiquement. Dans le cas du pro-rata, il faut indiquer à chaque agent quand s'arrêter de changer d'opinion. Pour cela, il faut établir un seuil *smax* de changements d'opinions à affecter à chaque agent. Pour chaque agent, dès qu'il a changé son opinion *smax* fois, il prendra aléatoirement une de ses opinions qu'il a eues parmi les *smax* opinions et il deviendra un agent têtu (ou explorateur) : il ne pourra plus changer d'opinion mais pourra toujours transmettre le sien. Dès que tous les agents deviennent des agents explorateurs, on considère que le consensus est atteint.

Variables	Signification
<i>n</i>	nombre initial d'agents explorateurs (têtus)
<i>nbreAgents</i>	nombre total d'agents
<i>smax</i>	seuil maximal de changements d'opinions
<i>nbreRessources</i>	nombre de ressources
<i>avisCourant[nbreRessources]</i>	liste des nombres d'opinions choisies
<i>indiceC</i>	indice de la ressource courante choisie

Algorithm 1: Algorithme minimal

```
while  $n \neq \text{nbreAgents}$  do
  forall agent disséminateur do
    while  $\text{changementsOpinions} \neq \text{smax}$  do
      Dissémination
      Voter model
       $\text{indiceC} = \text{indice de la ressource choisie par le Voter model}$ 
       $\text{changementsOpinions} = \text{changementsOpinions} + 1$ 
       $\text{avisCourant}[\text{indiceC}] = \text{avisCourant}[\text{indiceC}] + 1$ 
      Exploration
     $n = n + 1$ 
```

2.3.4 Résultats de l'algorithme minimal

On va mesurer l'évolution du nombre d'agents, et donc la qualité du consensus, au cours du temps. Les expérimentations se sont donc faites sur 50 simulations, avec 100 agents pour 2 ressources et 4 ressources, 99 agents pour 3 ressources avec différents rapports de valeurs de qualités pour chaque site. Puis sur ces 50 simulations, la moyenne pour chaque ressource a été faite. On rappelle que q_s est la qualité de la ressource s .

Pour deux ressources, on a :

FIGURE 2.2 – $q_a = 1, q_b = 2$

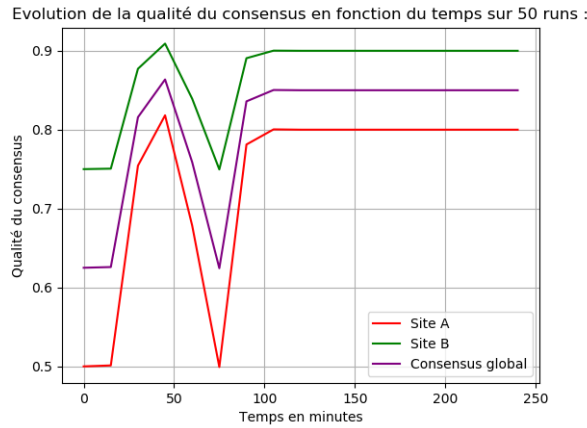


FIGURE 2.3 – $q_a = 1, q_b = 3$

Evolution de la qualité du consensus en fonction du temps sur 50 runs :

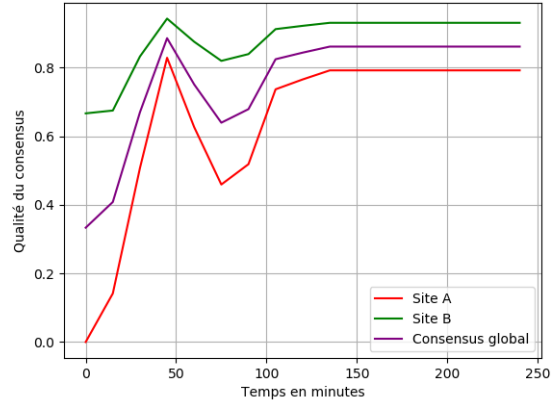
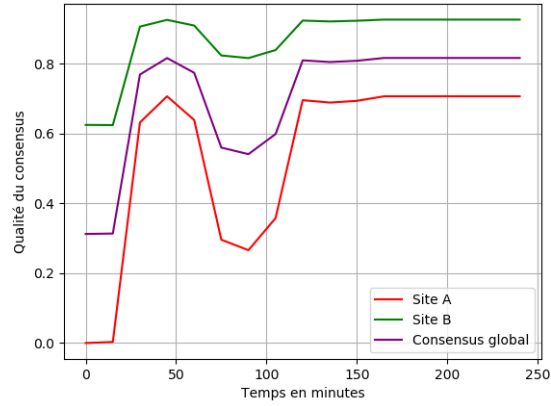


FIGURE 2.4 – $q_a = 1, q_b = 4$

Evolution de la qualité du consensus en fonction du temps sur 50 runs :



Lorsqu'il y a deux ressources, on constate que plus l'écart des qualités entre les deux ressources s'accroît, plus la qualité diminue.

Pour trois ressources, on a :

FIGURE 2.5 – $q_a = 1, q_b = 2, q_c = 3$

Evolution de la qualité du consensus en fonction du temps

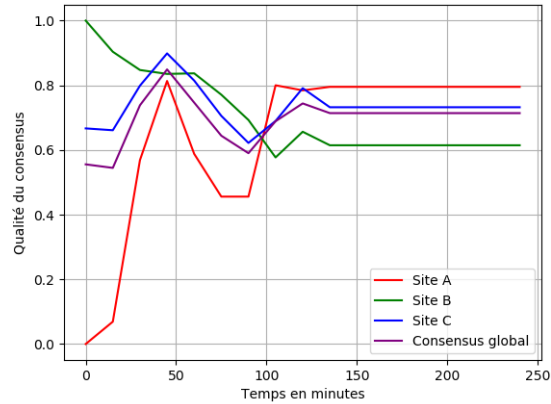
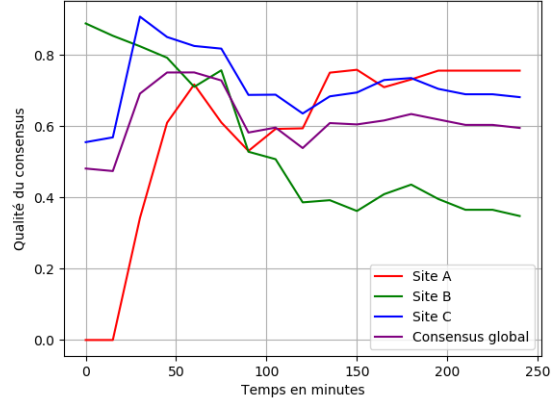


FIGURE 2.6 – $q_a = 1, q_b = 3, q_c = 6$

Evolution de la qualité du consensus en fonction du temps sur 50 runs :



On constate le même phénomène avec 3 ressources, la qualité du consensus global diminue avec l'augmentation du rapport entre les différentes qualités.

FIGURE 2.7 – $q_a = 1, q_b = 2, q_c = 3, q_d = 4$

Evolution de la qualité du consensus en fonction du temps sur 50 runs :

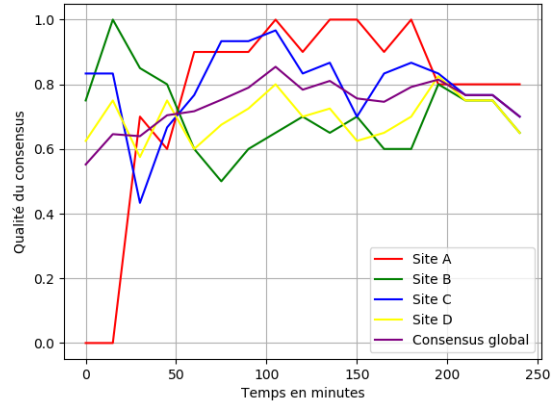
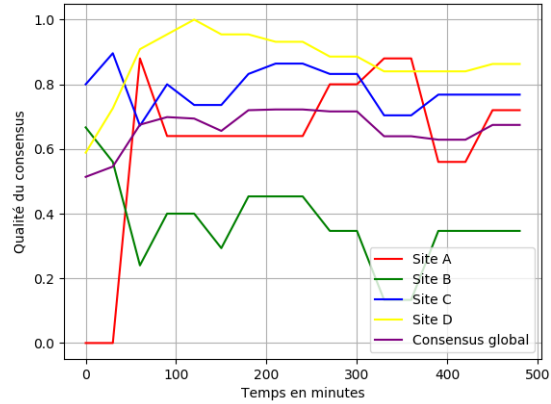


FIGURE 2.8 – $q_a = 1, q_b = 3, q_c = 5, q_d = 7$

Evolution de la qualité du consensus en fonction du temps sur 50 runs :



La qualité globale reste très médiocre, de l'ordre de 0,7, lorsqu'on possède autant de ressources.

Essayons d'améliorer en modifiant l'algorithme.

2.3.5 Algorithme amélioré

Variabes	Signification
n	nombre initial d'agents explorateurs (têtu)
$nbreAgents$	nombre total d'agents
$smax$	seuil maximal de changements d'opinions
$smin$	seuil minimal de changements d'opinions
$itCS$	itération critique pour le seuil dynamique
$poids$	poids à ajouter
$itCP$	itération critique pour le poids à ajouter
$nbreRessources$	nombre de ressources
$avisCourant[nbreRessources]$	liste des nombres d'opinions choisies
$indiceC$	indice de la ressource courante choisie
$qualiteC$	qualité de la ressource courante choisie

Algorithm 2: Algorithme "amélioré"

```

while  $n \neq nbreAgents$  do
  forall agent disséminateur do
    while  $changementsOpinions \neq smax$  do
       $changementsOpinions = changementsOpinions + 1$ 
       $avisCourant[indiceC] = avisCourant[indiceC] + 1$ 
      if  $changementsOpinions == itCP$  then
         $avisCourant[indiceC] = avisCourant[indiceC] + poids$ 
      if  $changementsOpinions == itCS$  then
         $smax = smin/qualiteC$ 
        if  $smax < changementsOpinions$  then
           $changementsOpinions = smax$ 
     $n++ = 1$ 

```
