SCIENCES SORBONNE UNIVERSITÉ

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

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 ou 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

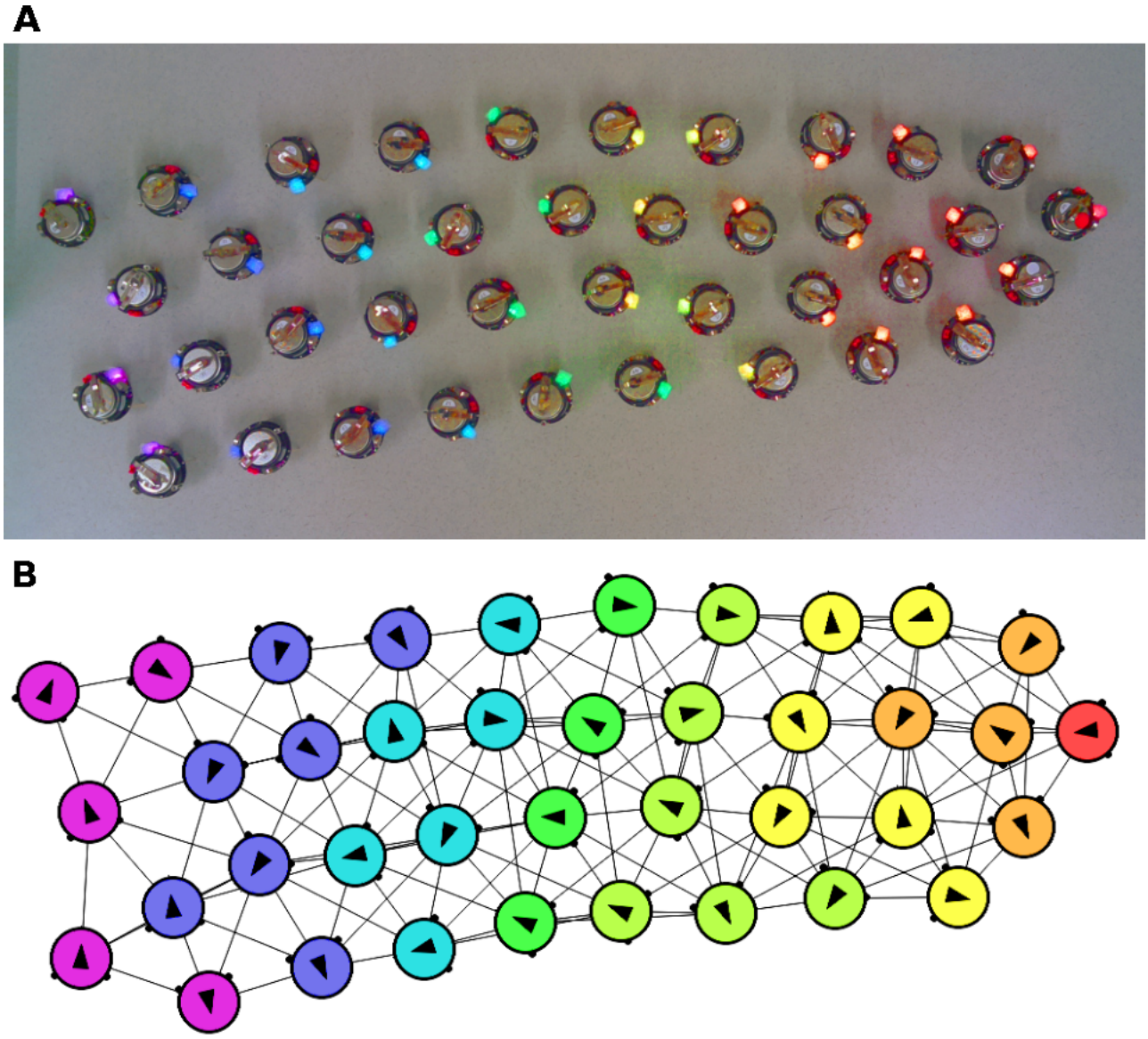
Les kilobots sont des mini robots possédant un diamètre de 33mm, destinés à la robotique en essaim. Ils ont été élaborés par l'Université d'Harvard dans le but de pouvoir appliquer des algorithmes utilisant des dizaines voire des centaines de robots, tout en réquérant un faible coût. Ils contiennent 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.

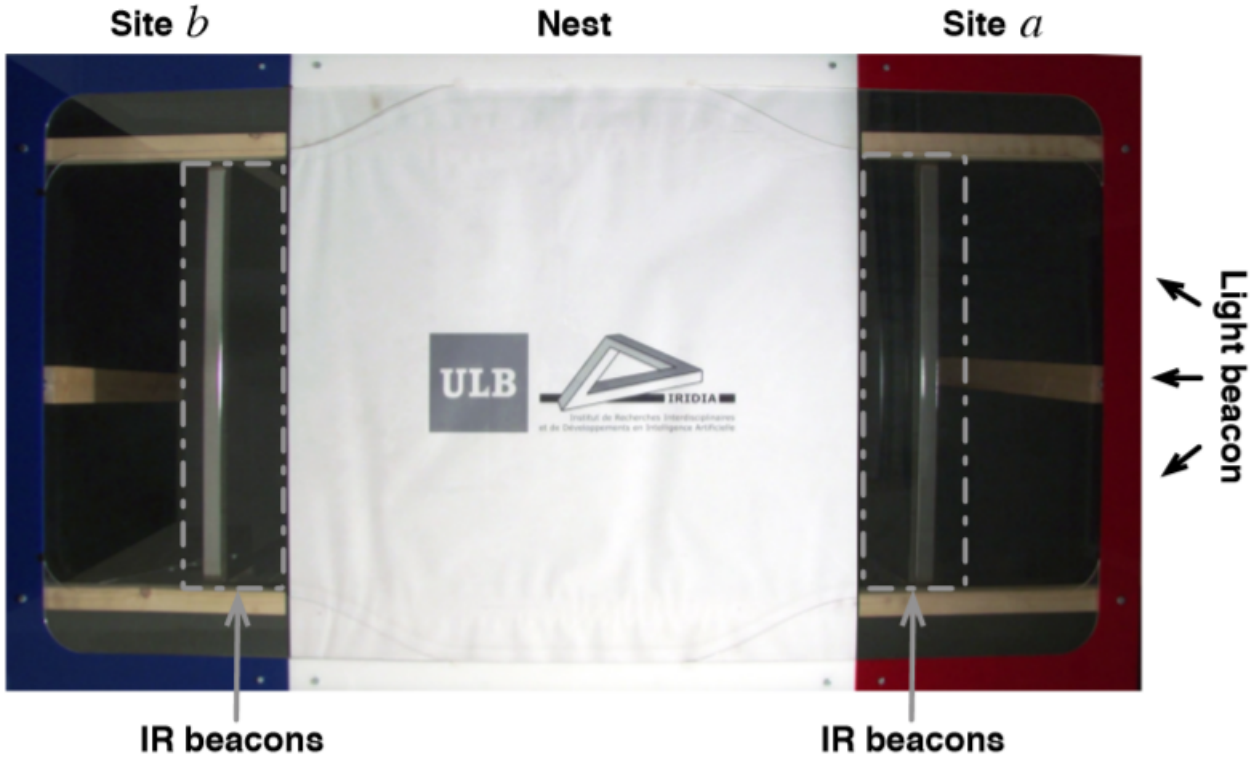
FIGURE 1.2 – Exemple d'un essaim de robots propageant un signal qui forme un gradient. (A) Le programme fonctionnant sur des Kilobots réels ; (B) Le même programme fonctionnant dans le simulateur. [4]



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 [2]. Il s'agit d'une arène rectangulaire de taille 100 x 190 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 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 cett 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

arene_kilombo.PNG

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 [2]. 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.

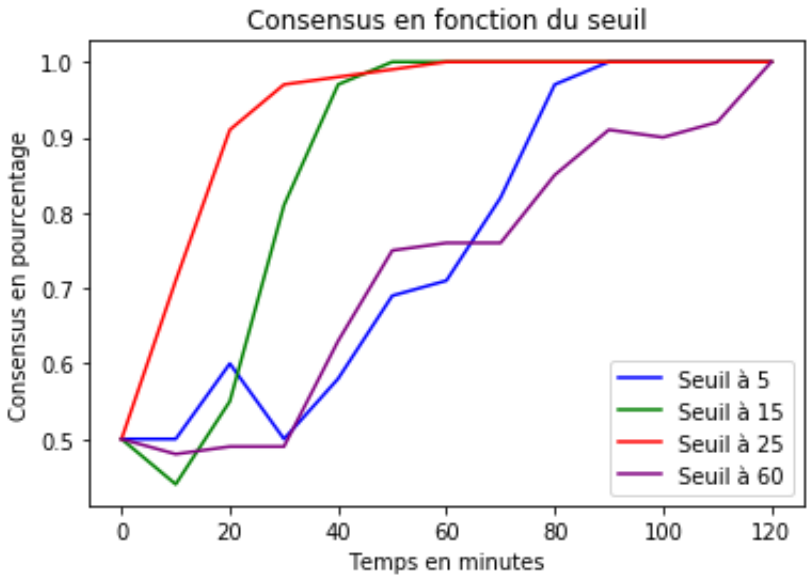
2.1.2 Voter model

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 [2] [5] avec les stratégies précédemment décrites, sachant que la meilleure est la stratégie du *Majority rule* [2]. L'objectif serait de réappliquer dans ce cas différents seuils décrits dans l'article [2] pour la stratégie du *Majority rule*, dans le cas où les qualites 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 inexitant 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 [2], 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 inviduellement 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 [6], 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 [6], 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 précdent 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. On définit par ailleurs pour la suite, qu'une itération pour un agent est un aller-retour nid-ressource, en d'autres termes un changement d'opinion.

Supposons $smax = 5$, et les chaangements d'opinions suivantes :

itération	1	2	3	4	5
Opinion courante	a	b	a	b	b

Dès qu'il aura fait 5 itérations alors il prendra une opinion parmi ce qu'il avait décidé auparavant, c'est-à-dire qu'il aura 40% de chance de prendre l'opinion a et 60% de chance de prendre l'opinion b jusqu'à que le consensus soit atteint.

Dès que tous les agents deviennent des agents explorateurs, on considère que le consensus est atteint.

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

Algorithm 1: Algorithme minimal

while $n \neq nbreAgents$ **do**
  forall *agent disséminateur* **do**
    while $changementsOpinions \neq smax$ **do**
      Dissémination
      Voter model
      $indiceC =$ *indice de la ressource choisie par le Voter model*
      $changementsOpinions = changementsOpinions + 1$
      $avisCourant[indiceC] = avisCourant[indiceC] + 1$
      Exploration
    L'agent devient un agent explorateur en faveur de la ressource prise aléatoirement parmi ses anciennes opinions.
    $n = n + 1$
  return *consensus*

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 avec $smax = 8$, on a :

FIGURE 2.2 – $q_a = 1$, $q_b = 2$

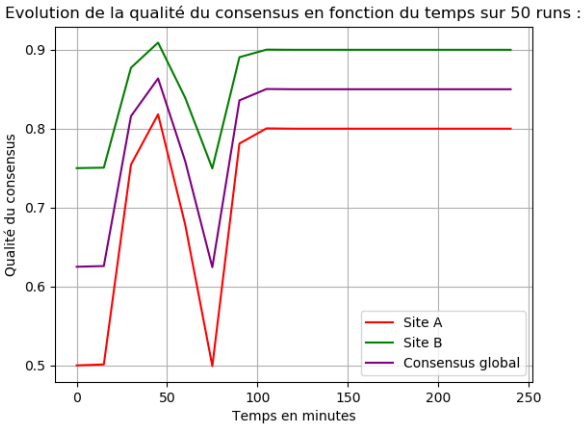


FIGURE 2.3 – $q_a = 1$, $q_b = 3$

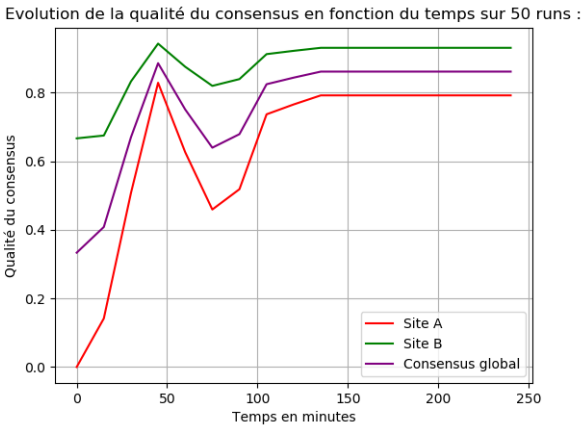
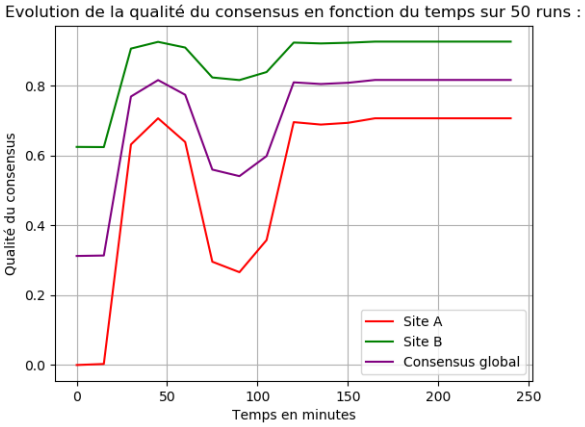


FIGURE 2.4 – $q_a = 1$, $q_b = 4$



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

Pour trois ressources avec $smax = 8$, on a :

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

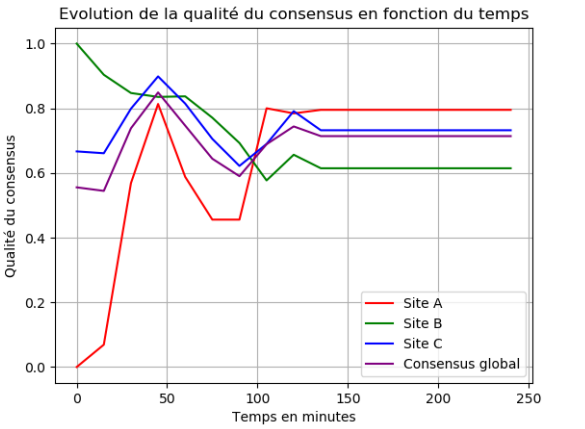
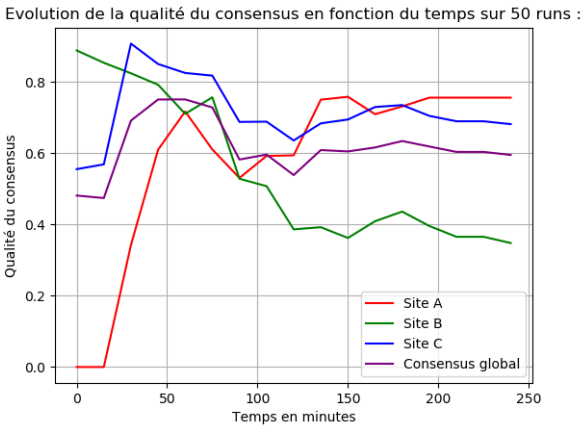


FIGURE 2.6 – $q_a = 1$, $q_b = 3$, $q_c = 6$



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

Pour quatre ressources avec $smax = 12$, on a :

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

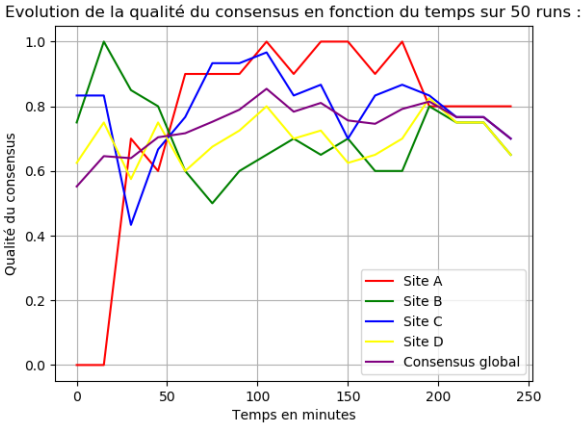
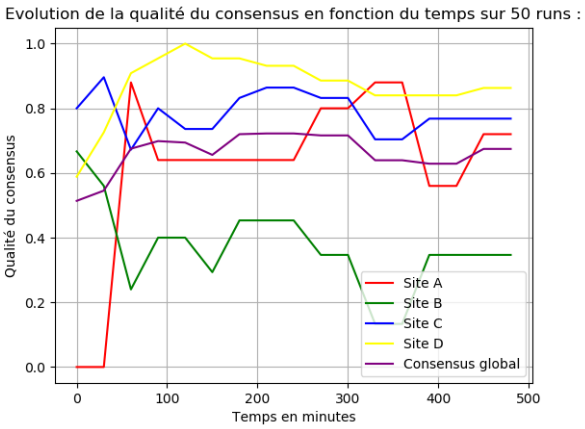


FIGURE 2.8 – $q_a = 1$, $q_b = 3$, $q_c = 5, q_c = 7$



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. D'une manière générale, on constate qu'à certains moments des runs, en analysant l'évolution de la qualité du consensus au cours du temps dans les graphes précédents, le nombre d'agents ayant l'opinion optimale est meilleur. Par exemple pour deux ressources, à 40 minutes sur leurs graphes, le nombre d'agents possédant l'opinion optimale permettant le pro-rata a une qualité meilleure que celle du consensus atteint final. Il existerait donc une itération où si les agents garderaient leur avis courant, alors le consensus serait meilleur.

Il serait aussi intéressant de modifier le seuil de base $smax$, le modifier de sorte qu'il devienne un seuil dynamique qui pourrait se modifier au cours de la simulation.

Autre problème aperçu lors des simulations individuelles, il était possible d'avoir un énorme déquilibre entre les ressources de meilleure et moindre qualité. Par exemple pour le cas de $q_a = 1$ et $q_b = 2$, il pouvait avoir seulement 10 au lieu de 34 agents pour le site a. Cela est dû à l'aspect aléatoire du *Voter model*. Il faut donc ajouter plus d'influence pour les sites possédant une qualité faible.
Pour cela, on peut comme Judhi Prasetyo [6], mettre n agents têtus dès le début. Il y aurait alors n divisé par le nombre de ressources, agents têtus pour chaque ressource dès le commencement.

2.3.5 Algorithme amélioré

Il s'agit d'essayer d'améliorer le processus précédent en ajoutant deux nouvelles fonctions à l'algorithme : rajouter un poids à l'itération où le nombre d'agents expérimental est le plus proche du nombre d'agents théorique, et modifier le seuil au cours du temps pour essayer de mieux l'adapter.

Dans ce cas l'objectif serait de trouver les meilleures itérations pour ça.

Ajouter un poids à une itération signifie qu'à cette itération critique, l'opinion compte une fois (comme dans l'algorithme précédent) mais en plus le poids.

Par exemple si on ajoute un poids de 1 à l'itération 2, on aurait :

itération	1	2	3	4	5
Opinion courante	a	b	b	b	b

On ajouterait la même opinion dans notre liste en plus, d'où le fait qu'on compte l'opinion b pour l'itération 3 (c'est l'effet du poids).

Modifier le seuil au cours du temps consiste à faire devenir un agent disséminateur un agent explorateur plus tôt que prévu en fonction d'une opinion à une itération précise, qui serait critique aussi. Cela permet d'ajouter une influence en faveur de la ressource choisie à cette itération. En effet, cela signifierait que parmi les $smax$ ressources (ou opinions) que l'agent a visitées, il y aurait au moins une opinion en faveur de cette ressource.
Soit $smin$ un autre seuil, et q_s la valeur de la qualité de la ressource s choisie à l'itération critique :

$$smax = \frac{smin}{q_s}$$

Choisissons arbitrairement $smin = smax$. Cette formule donnerait alors cet aspect : plus la qualité de la ressource choisie à cette itération critique, est faible, plus l'agent deviendra un agent explorateur rapidement. Et inversement plus la qualité de cette ressource est élevée, plus l'agent prendra des itérations avant d'arrêter de changer d'avis.
Cela permet d'ajouter davantage d'influence pour les agents pouvant choisir les ressources ayant

une faible qualité, afin de pallier au problème du déséquilibre du nombre d'agents entre la meilleure ressource et la moindre aperçu précédemment.

Variables	Signification
n	nombre initial d'agents explorateurs (têtus)
$nbreAgents$	nombre total d'agents
$smax$	seuil maximal de changements d'opinions
$smin$	seuil mimimal 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 ≠ nbreAgents do
    forall agent disséminateur do
        while changementsOpinions ≠ 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
```

2.3.6 Résultats de l'algorithme amélioré

Tout d'abord prenons les cas avec deux ressources.

Le premier graphe correspond à ces valeurs :

Variables	Valeur
n	6
$nbreAgents$	100
$smax$	8
$smin$	0
$itCS$	0
$poids$	1
$itCP$	3
$nbreRessources$	2

Le second graphe correspond à ces valeurs :

Variables	Valeur
n	6
$nbreAgents$	100
$smax$	8
$smin$	8
$itCS$	2
$poids$	0
$itCP$	0
$nbreRessources$	2

Le troisième graphe correspond à ces valeurs :

Variables	Valeur
n	6
$nbreAgents$	100
$smax$	8
$smin$	8
$itCS$	2
$poids$	1
$itCP$	3
$nbreRessources$	2

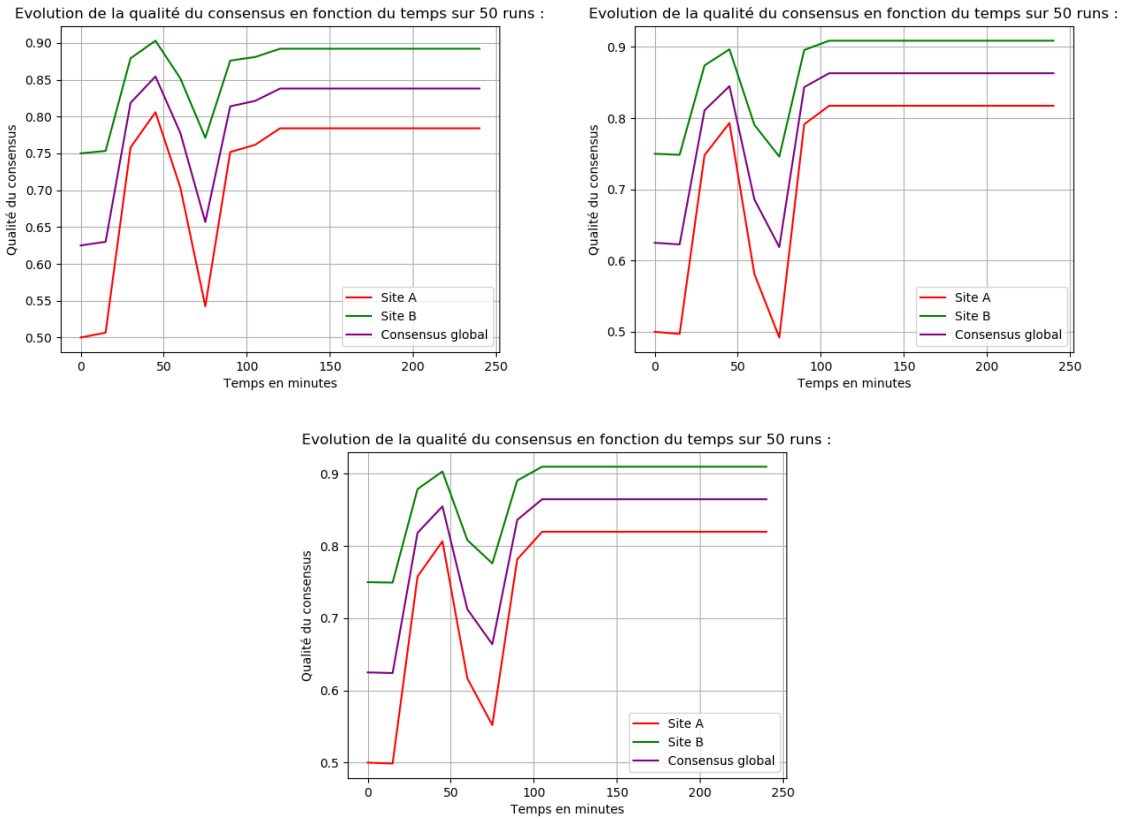


FIGURE 2.9 – $q_a = 1$, $q_b = 2$

On constate alors une légère amélioration par rapport à l'algorithme minimal lorsqu'on active au moins le paramètre du poids à ajouter (deuxième graphe et troisième graphe).

Dans le cas avec $q_a = 1$ et $q_b = 3$ ou $q_b = 4$, on constate qu'il est préférable d'utiliser l'algorithme minimal et non amélioré.

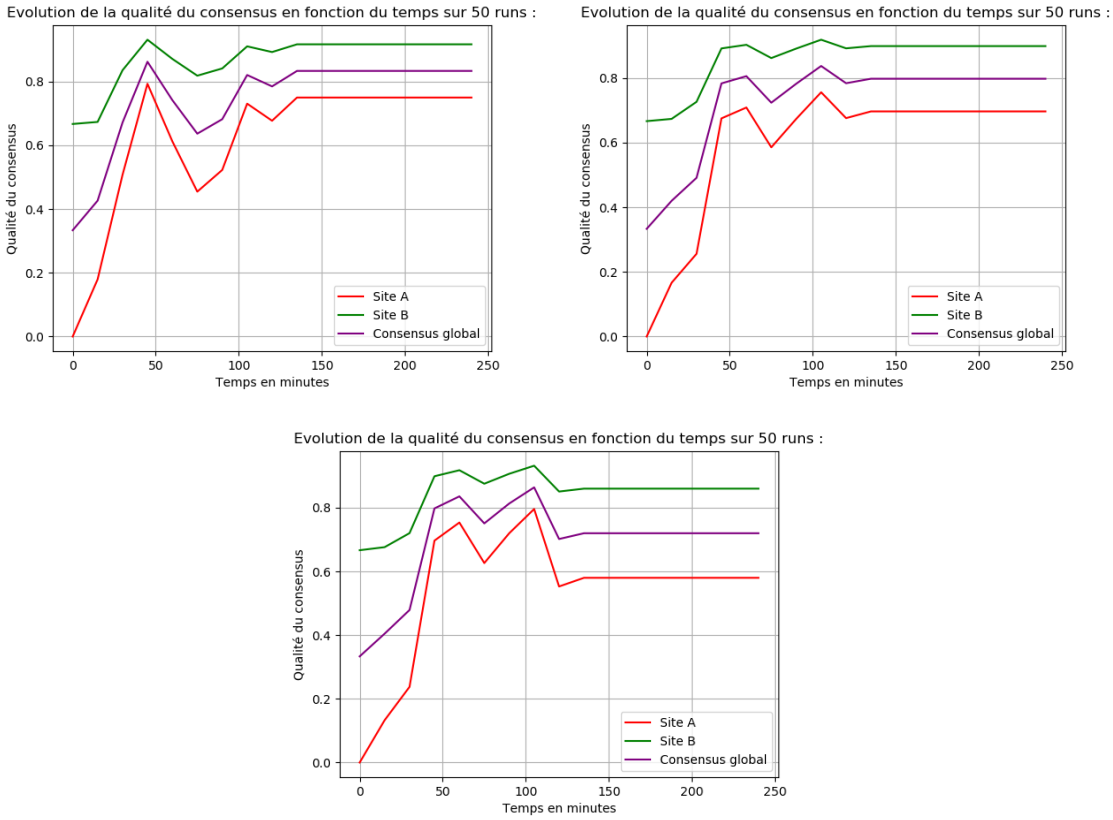


FIGURE 2.10 – $q_a = 1$, $q_b = 3$

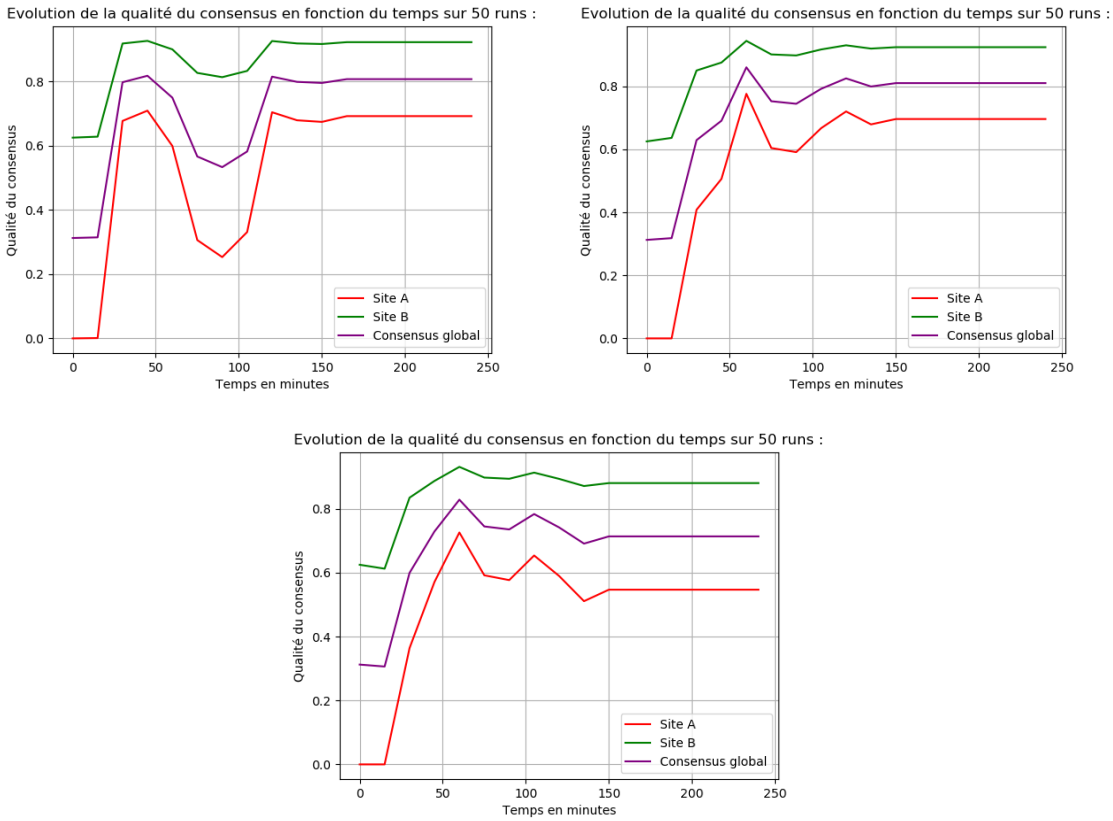


FIGURE 2.11 – $q_a = 1$, $q_b = 4$

Prenons les cas avec trois ressources.

Le premier graphe correspond à ces valeurs :

Variables	Valeur
n	12
$nbreAgents$	99
$smax$	8
$smin$	0
$itCS$	0
$poids$	1
$itCP$	3
$nbreRessources$	3

Le second graphe correspond à ces valeurs :

Variables	Valeur
n	12
$nbreAgents$	99
$smax$	8
$smin$	8
$itCS$	2
$poids$	0
$itCP$	0
$nbreRessources$	3

Le troisième graphe correspond à ces valeurs :

Variables	Valeur
n	12
$nbreAgents$	99
$smax$	8
$smin$	8
$itCS$	2
$poids$	1
$itCP$	3
$nbreRessources$	3

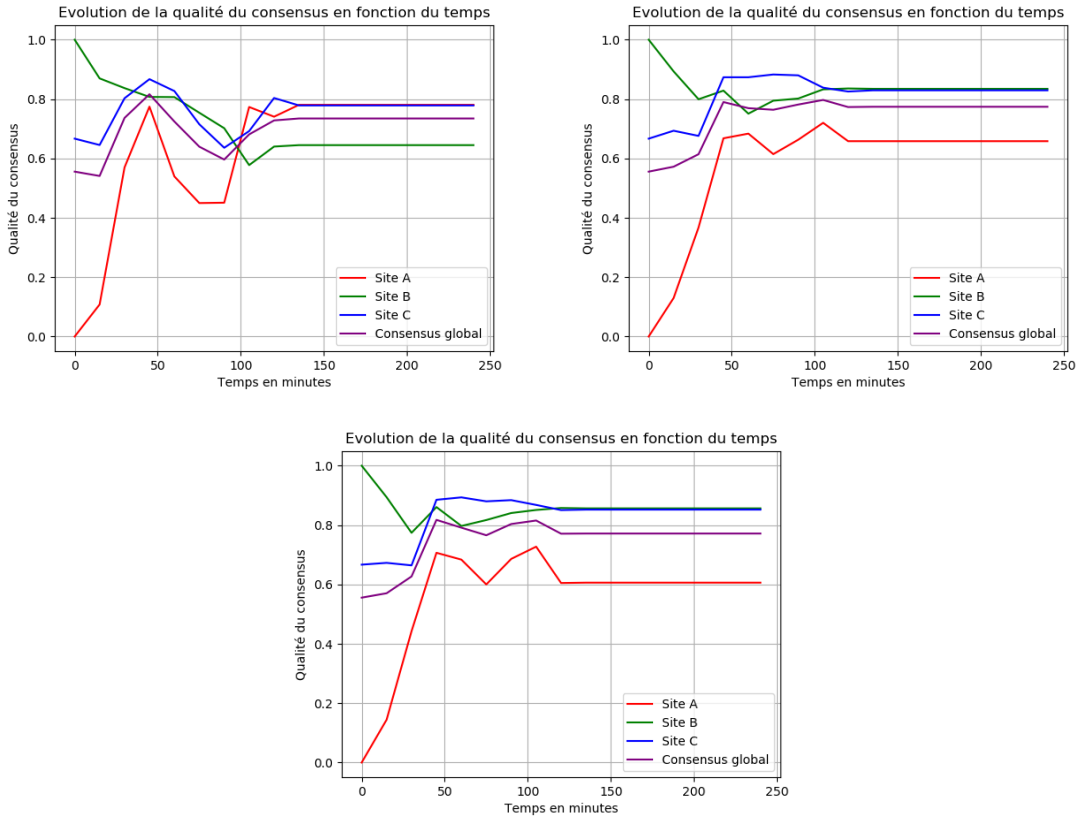


FIGURE 2.12 – $q_a = 1$, $q_b = 2$, $q_c = 3$

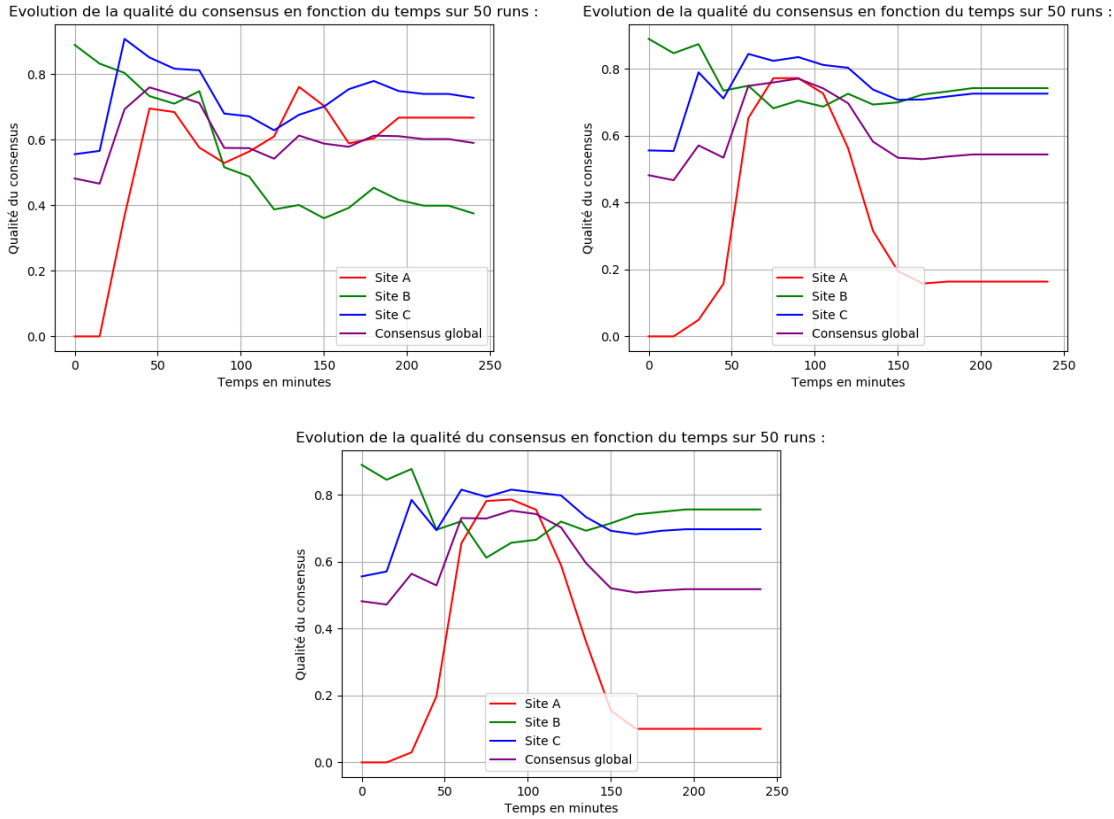


FIGURE 2.13 – $q_a = 1$, $q_b = 3$, $q_c = 6$

On constate que dans un premier temps, avec un rapport de qualité faible entre les qualités des ressources, l'algorithme amélioré est meilleur que l'algorithme minimal. En revanche, lorsque l'écart entre ces différentes valeurs est prononcé, l'algorithme amélioré est moins efficace que l'algorithme de base.

Prenons le cas avec 4 ressources.

Le premier graphe correspond à ces valeurs :

Variables	Valeur
n	16
$nbreAgents$	100
$smax$	12
$smin$	0
$itCS$	0
$poids$	2
$itCP$	1
$nbreRessources$	4

Le second graphe correspond à ces valeurs :

Variables	Valeur
n	16
$nbreAgents$	100
$smax$	12
$smin$	12
$itCS$	2
$poids$	0
$itCP$	0
$nbreRessources$	4

Le troisième graphe correspond à ces valeurs :

Variables	Valeur
n	16
$nbreAgents$	100
$smax$	12
$smin$	12
$itCS$	2
$poids$	2
$itCP$	3
$nbreRessources$	4

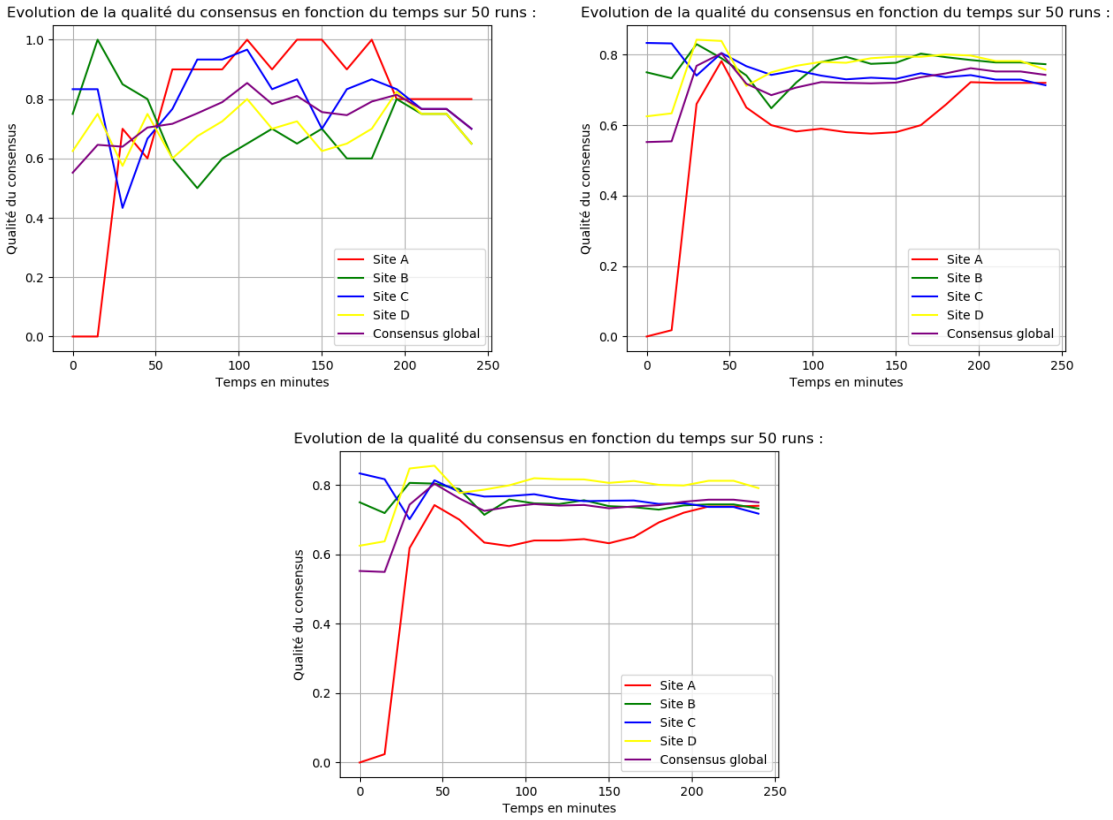


FIGURE 2.14 – $q_a = 1$, $q_b = 2$, $q_c = 3$, $q_d = 4$

Le consensus global est bien meilleur avec l'algorithme amélioré.

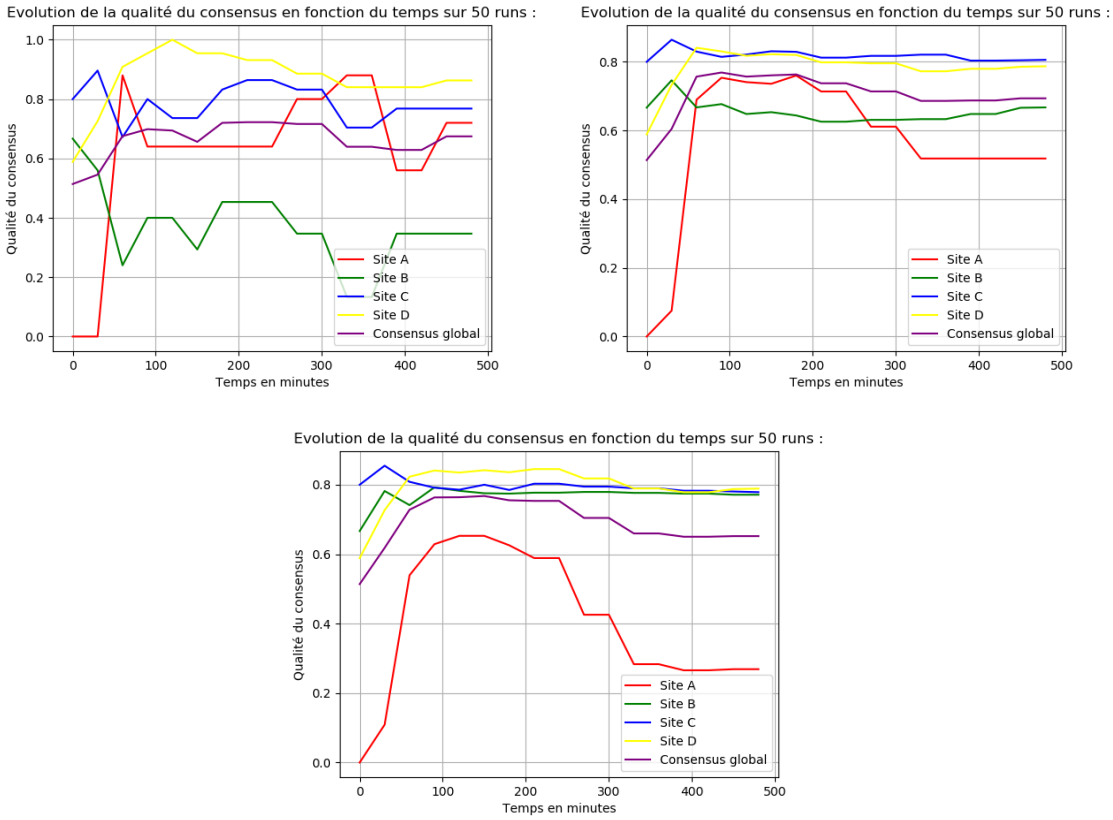


FIGURE 2.15 – $q_a = 1$, $q_b = 3$, $q_c = 5$, $q_d = 7$

La meilleure méthode dans ce cas serait d'utiliser seulement le poids à ajouter, et de ne pas utiliser la modification du seuil. Malgré cela, il reste assez équivalent à l'algorithme minimal et ne l'améliore pas.

Chapitre 3

L'algorithme d'apprentissage

Bibliographie

[1] Gabriele Valentini, Eliseo Ferrante, and Marco Dorigo. *The Best-of-n Problem in Robot Swarms : Formalization, State of the Art, and Novel Perspectives*. Front. Robot. AI 4, (2017). DOI :https ://doi.org/10.3389/frobt.2017.00009

[2] Gabriele Valentini, Eliseo Ferrante, Heiko Hamann, and Marco Dorigo. *Collective decision with 100 Kilobots : speed versus accuracy in binary discrimination problems*. Autonomous Agents and Multi-Agent Systems 30, 3 (2016), 553–580. DOI :https ://doi.org/10.1007/s10458-015-9323-3

[3] Nicolas Bredeche, Jean-Marc Montanier, Wenguo Liu, and Alan F.T. Winfield. *Environment-driven distributed evolutionary adaptation in a population of autonomous robotic agents*. Mathematical and Computer Modelling of Dynamical Systems 18, 1 (February 2012), 101–129. DOI :https ://doi.org/10.1080/13873954.2011.601425

[4] Fredrik Jansson, Matthew Hartley, Martin Hinsch, Ivica Slavkov, Noemí Carranza, Tjelvar S. G. Olsson, Roland M. Dries, Johanna H. Grönqvist, Athanasius F. M. Marée, James Sharpe, Jaap A. Kaandorp, and Verônica A. Grieneisen. *Kilombo : a Kilobot simulator to enable effective research in swarm robotics.*. arXiv :1511.04285 [cs] (May 2016). http ://arxiv.org/abs/1511.04285

[5] Gabriele Valentini, Heiko Hamann, and Marco Dorigo. *Self-organized Collective Decision Making : The Weighted Voter Model.*. 2014.

[6] Judhi Prasetyo, Giulia De Masi, Pallavi Ranjan, and Eliseo Ferrante. *The Best-of-n Problem with Dynamic Site Qualities : Achieving Adaptability with Stubborn Individuals*. Swarm Intelligence 5, 3 (2011), 305–327. 11th International Conference, ANTS 2018, Rome, Italy, October 29–31, 2018, Proceedings. . 239–251.