



MODÈLE MULTI-AGENTS DE RATIONALITÉ
DANS LES ORGANISATIONS
ÉMERGENCE D'UNE CULTURE D'ENTREPRISE

PROJET MOSIMA 1

Etudiants :
Fayçal EL OUARIACHI
Thibault GIGANT

Encadrant :
Jean-Daniel KANT

Année 2016 - 2017

Table des matières

1 Etude du modèle	1
1.1 Corporate culture	1
1.2 Attributs et méthodes	2
1.3 Formules et Algorithmes du modèle	2
2 Implémentation	6
2.1 Interface	6
2.2 L'affichage des figures	7
2.3 Informations notables sur l'implémentation	8
2.3.1 Initialisation	8
2.3.2 Implémentation de la mise à jour de l'effort des agents rationnels	8
3 Expérimentations et reproduction des résultats	8
3.1 Émergence d'une corporate culture	8
3.2 Absence de convergence	9
3.3 L'impact des agents à effort élevé	10
3.3.1 Reproduction de la figure 6 de l'article	10
3.3.2 Reproduction de la figure 7 de l'article	11
3.4 L'impact du bruit	11
3.5 Suggestions d'améliorations du modèle	12
3.6 Implémentation d'une amélioration	13
3.6.1 Implémentation	13
3.6.2 Résultats	13
Conclusion	14
A Annexes	15
A.1 Figures	17
A.2 Bibliographie	38

1 Etude du modèle

1.1 Corporate culture

L'article [1], sujet de cette étude, tente d'étudier l'évolution de ce que les auteurs définissent comme la « corporate culture », ce qu'on pourrait traduire par « culture d'entreprise », en fonction de la composition de l'entreprise. Il est donc important de définir ce qu'est cette notion. Les auteurs font référence à de multiples définitions, dont une idée de Kreps [2], qui voit la *corporate culture* comme un ensemble de principes que tout le monde est sensé suivre. Cependant, DAL FORNO et MERLONE ont décidé de l'étudier comme étant un niveau d'effort commun qui est exercé par l'ensemble des employés.

Ils ne considèrent aucune tâche particulière, juste l'effort promulgué par les agents, qui sont les membres de l'entreprise, pour accomplir cette tâche. Lorsque deux agents se font face, ils forment une équipe où chacun fournit un effort, dans le but d'effectuer cette tâche indéfinie. Plusieurs types d'agents ont été définis dans l'étude, chacun symbolisant un style de comportement observé dans les entreprises. Ces comportements diffèrent par l'adaptation de leur effort face à l'effort qu'ils observent chez leurs partenaires.

Les auteurs définissent une fonction sensée représenter le profit retiré d'une collaboration avec un autre agent. Cette fonction dépend évidemment de l'effort qu'un agent exerce, mais aussi de l'effort de son partenaire.

Chaque agent, durant une collaboration, peut évaluer le profit qu'apporte son travail, en fonction de l'effort que lui-même et son partenaire fournissent.

Les différents types d'agents étudiés dans cet article, ainsi que l'algorithme ou l'équation qu'ils suivent pour mettre à jour leur effort, sont les suivants :

- *null effort* : Ces agents exercent toujours le même effort très bas. En l'occurrence, cet effort a été réglé arbitrairement à 0.0001 dans l'article.
Cet agent ne met jamais à jour son effort.
- *shrinking effort* : Ces agents fournissent pour effort la moitié de l'effort de l'agent qu'ils ont précédemment rencontré. Il commencent d'abord avec un effort tiré aléatoirement.
- *replicator* : Ces agents copient l'effort qu'ils perçoivent de leur partenaire pour l'exercer au tour suivant. Initialement, ils commencent avec un effort aléatoire.
- *rational* : Ces agents maximisent le profit qu'ils vont engendrer, en prenant en compte l'effort que son partenaire avait fourni à sa dernière collaboration. L'algorithme utilisé pour cet agent sera détaillé par la suite, car il nécessite quelques calculs à préciser.
- *profit comparator* : Ceux-ci comparent leur profit à celui qu'ils perçoivent de leur partenaire. Si le leur est plus élevé, alors ils augmentent de 10% l'effort qu'ils exercent au prochain tour. Sinon ils le diminuent de 10%.
- *high effort* : Ces agents exercent toujours le même effort élevé. En l'occurrence, cet effort a été réglé arbitrairement à 2.001 dans l'article.
Cet agent ne met jamais à jour son effort.
- *average rational* : Ces agents maximisent le profit qu'ils vont engendrer, en prenant en compte l'effort moyen de l'ensemble des partenaires qu'ils avaient rencontrés.
- *winner imitator* : Ceux-ci commencent avec un effort élevé, arbitrairement similaire à celui des « *high effort* », puis copient l'effort qu'ils perçoivent de leur partenaire si celui-ci se révèle donner un meilleur profit.
- *effort comparator* : Ceux-ci comparent leur effort à celui qu'ils perçoivent de leur partenaire. Si le leur est plus élevé, alors ils augmentent de 10% l'effort qu'ils exercent au prochain tour. Sinon ils le diminuent de 10%.
- *averager* : Ces derniers commencent avec un effort aléatoire, puis à chaque rencontre modifient l'effort qu'ils exercent en prenant la moyenne des efforts exercés par lui-même et son partenaire.

D'après la modélisation proposée par l'article, les agents ne peuvent pas changer de type. De plus, lorsqu'ils évaluent l'effort fourni par leur partenaire, ils peuvent commettre une erreur. Cette erreur est représentée par un « bruit » appliquée lors de la récupération de l'effort de l'agent partenaire.

L'environnement se présente sous la forme d'une grille, dont les cellules ne peuvent contenir qu'un seul agent à la fois. Il est considéré comme toroïdal, c'est-à-dire que les agents peuvent « traverser » un bord pour arriver de l'autre côté.

1.2 Attributs et méthodes

Afin de pouvoir mener nos simulations à bien, chaque agent devra garder en mémoire certaines données. En plus de son type, on gardera l'effort qu'il a fourni (**leffort**), celui qu'il devra fournir après chaque collaboration (**effort**), et du profit obtenu (**profit**). Chacun gardera une trace des effort et profit du dernier collaborateur rencontré (**aeffort** et **aprofit**), ainsi qu'un moyen de calculer la moyenne sur l'ensemble des agents avec qui il a travaillé. Il faut donc garder des attributs permettant d'effectuer ce calcul : la somme des efforts des partenaires (**cumeffort**), ainsi que le nombre de partenaires que l'on a eus jusqu'à présent (**numinc**).

En ce qui concerne les déplacements, un attribut donnant la direction de l'agent (**dir**), et un autre donnant le pas autorisé (**step**) sont nécessaires. Pour la visualisation, des attributs correspondant aux coordonnées (**posx** et **posy**), et aux couleurs représentant le type de l'agent (**colt**) et son effort (**col**), seront aussi attribués.

Quelques méthodes principales devront être possédées par chaque agent :

- *RandAgent* : Servira à initialiser un agent dans l'environnement. Il devra être placé sur la grille, dans une direction donnée, et avec un effort correspondant à son type.
- *RandMove* : Servira à déplacer un agent, dans la direction où il regarde, d'une case, si celle-ci est libre.
- *WorkAgent* : Servira à entamer une procédure de collaboration entre deux agents se faisant face.
- *RandDir* : Servira à changer l'orientation d'un agent vers l'un des quatre points cardinaux.

1.3 Formules et Algorithmes du modèle

Chaque agent calcule son propre profit lors d'une collaboration, selon une formule donnée dans l'article :

$$\pi_i = 5\sqrt{e_i + e_j} - e_i^2 \quad (1)$$

où π_i est le profit de l'agent, e_i est son effort et e_j celui de son partenaire.

Quant au bruit, lorsqu'on le prend en compte, il est appliquée lors de la détection de l'effort fourni par le partenaire, on choisit de manière aléatoire uniforme cet effort dans l'intervalle suivant :

$$[\text{aeffort} \times (1 - bruit), \text{aeffort} \times (1 + bruit)]$$

où *bruit* est un flottant entre 1% et 50% (donc entre 0.01 et 0.50) paramétré à l'avance.

La simulation suivra un schéma d'exécution simple. Après une initialisation de l'environnement et de ses agents, et donc des appels à la méthode *RandAgent*, ainsi que le rendu visuel de cet environnement, à chaque pas de temps les agents exécuteront dans l'ordre les actions suivantes :

1. *Bouger* d'un pas, suivant l'un des quatre points cardinaux choisi aléatoirement, si l'espace n'est pas déjà occupée. Cette phase correspond à la méthode *RandMove*.
2. *Collaborer* avec un autre agent, si quelqu'un lui fait face. Chacun exercera son propre effort lors de cette collaboration. Cette phase correspond à l'appel de la méthode *WorkAgent*.

3. *Adapter* leur comportement en fonction de l'expérience qu'ils viennent d'avoir. C'est à cette étape qu'ils mettent à jour l'effort qu'ils fourniront à l'itération suivante.

On pourrait résumer cela par cet algorithme simplifié :

Algorithme 1 : Algorithme d'exécution de la simulation

```

1 début
2   pour chaque agent  $i$  faire
3     | RandMove
4   fin
5   pour chaque agent  $i$  faire
6     | si Partenaire( $i$ ) alors
7       |   WorkAgent
8     | fin
9   fin
10  pour chaque agent  $i$  faire
11    | Adapt
12  fin
13 fin

```

où « $\text{Partenaire}(i)$ » vaut vrai si l'*agent i* fait face à un autre agent.

Le reste des algorithmes du modèle est concentré dans la mise à jour de l'effort de chaque type d'agent.

Pour un agent *shrinking effort* :

Algorithme 2 : Mise à jour de l'effort d'un Shrinking Effort

Entrées :

Effort du partenaire **parteff**

```

1 début
2   | effort = parteff / 2
3 fin

```

Pour un agent *replicator* :

Algorithme 3 : Mise à jour de l'effort d'un Replicator

Entrées :

Effort du partenaire **parteff**

```

1 début
2   | effort = parteff
3 fin

```

Pour un agent *profit comparator* :

Algorithme 4 : Mise à jour de l'effort d'un Profit Comparator

Entrées :Effort du partenaire `parteff`

```

1 début
2   profit = calculate-profit(effort, parteff);; Profit de l'agent
3   aprofit = calculate-profit(parteff, effort);; Profit du partenaire
4   si profit > aprofit alors
5     | effort = effort × 1.1
6   fin
7   sinon
8     | effort = effort × 0.9
9   fin
10 fin

```

où `calculate-profit` est une méthode donnant le profit de l'agent dont l'effort est passé en premier paramètre, et l'effort du partenaire en second.

Pour un agent *winner imitator* :

Algorithme 5 : Mise à jour de l'effort d'un Winner Imitator

Entrées :Effort du partenaire `parteff`

```

1 début
2   profit = calculate-profit(effort, parteff);; Profit de l'agent
3   aprofit = calculate-profit(parteff, effort);; Profit du partenaire
4   si profit < aprofit alors
5     | effort = parteff
6   fin
7 fin

```

Pour un agent *effort comparator* :

Algorithme 6 : Mise à jour de l'effort d'un Effort Comparator

Entrées :Effort du partenaire `parteff`

```

1 début
2   si effort > parteff alors
3     | effort = effort × 1.1
4   fin
5   sinon
6     | effort = effort × 0.9
7   fin
8 fin

```

Pour un agent *averager* :

Algorithme 7 : Mise à jour de l'effort d'un Averager**Entrées :**Effort du partenaire **parteff**

```

1 début
2   |   effort = (effort + parteff) / 2
3 fin

```

Les agents rationnels sont particuliers. Ils demandent un calcul relativement compliqué par rapport aux autres types d'agents. Après chaque rencontre, ils exercent l'effort qui aurait servi à maximiser leur profit :

Algorithme 8 : Mise à jour de l'effort d'un Rational**Entrées :**Effort du partenaire **parteff**

```

1 début
2   |   effort = argmaxeffort 5 √effort + parteff - effort2
3 fin

```

Le même algorithme est appliqué aux agent Average Rational, à la seule différence que la variable **parteff** ne représente plus la valeur de l'effort du partenaire, mais de la moyenne des efforts des agents rencontrés jusqu'à présent. Cette moyenne est calculée en prenant le rapport entre la somme des efforts des partenaires passés **cumeffort** et le nombre de ces rencontres **numinc**.

Précisions les calculs nécessaires pour trouver cet argmax. On veut maximiser π_i (Formule 1). Pour cela, étudions sa dérivée par rapport à e_i , que nous nommerons π'_i , et sa dérivée seconde, que nous nommerons π''_i . On a donc :

$$\pi'_i = \frac{5}{2\sqrt{e_i + e_j}} - 2e_i \quad (2)$$

$$\pi''_i = -\frac{5}{4\sqrt{e_i + e_j}^3} - 2 \quad (3)$$

Comme on considère que les efforts sont toujours positifs, on en conclut que $\pi''_i < 0$ car $e_i > 0$ et $e_j > 0$. Donc π_i est une fonction concave. Pour trouver son maximum, il suffit de trouver e_i qui annule π'_i .

$$\pi'_i = 0 \iff \frac{5}{2\sqrt{e_i + e_j}} - 2e_i = 0 \iff e_i\sqrt{e_i + e_j} = \frac{5}{4} \iff e_i^3 + e_j e_i^2 - \frac{25}{16} = 0 \quad (4)$$

On a donc une équation cubique de la forme $ax^3 + bx^2 + cx + d = 0$ avec $a = 1, b = e_j, c = 0, d = -\frac{25}{16}$. Nous avons utilisé la méthode de Cardan pour la résoudre. Posons :

$$\begin{cases} p = \frac{3ac - b^2}{3a^2} = -\frac{e_j^2}{3} \\ q = \frac{27a^2d - 9abc + 2b^3}{27a^3} = \frac{2}{27}e_j^3 - \frac{25}{16} \end{cases}$$

De plus, on calcule un discriminant :

$$\Delta = \frac{q^2}{4} + \frac{p^3}{27} = \frac{625}{1024} - \frac{25}{432}e_j^3$$

En considérant que $e_j \in [0.0001, 2.001]$, on obtient que ce discriminant Δ est toujours positif. Pour qu'il ne le soit pas, e_j ne doit pas dépasser $\frac{625 \times 432}{1024 \times 25} = 10.546875$, or les valeurs d'effort prises dans l'article sont toujours comprises entre 0.0001 et 2.001.

L'équation 4 a donc une unique solution réelle donnée par la formule suivante :

$$e_i = \sqrt[3]{-\frac{q}{2} + \sqrt{\Delta}} + \sqrt[3]{-\frac{q}{2} - \sqrt{\Delta}} - \frac{b}{3a} \quad (5)$$

La démonstration de cette méthode est disponible sur le site d'Alain LASSINE [3].

2 Implémentation

2.1 Interface

Dans le but de vérifier les figures et résultats donnés par l'article, nous avons implémenté en NetLogo un moyen d'effectuer les simulations nécessaires. Pour cela, il faut tout d'abord déterminer les paramètres que l'on pourra régler. Chacun doit pouvoir être changé directement dans l'interface. Les paramètres sont les suivants :

- La taille de l'environnement, à rentrer dans l'onglet « *Settings...* » de l'interface.
- Le nombre d'agents de chaque type. Une valeur est demandée pour chaque type d'agent, sous la forme d'un *input* NetLogo dont la valeur est associée à la variable `nb-agents-<type>` (où `<type>` est bien entendu à remplacer par le type de l'agent dont le nombre est spécifié).
- La taille des pas que peuvent effectuer les agents à chaque itération. Cette variable est à régler dans l'interface par un *Slider*.
- L'activation ou non de l'affichage de l'effort. Cette possibilité est donnée par un *Switch* relié à la variable `display-effort`.
- L'activation ou non du bruit lors de la simulation. Là aussi, un *Switch* permet de l'activer. De plus, la valeur du bruit est paramétrable par un *Slider* relié à la variable `noise-value`, allant de 1% à 50%.

La figure 1 montre la portion de l'interface permettant de moduler les paramètres précédemment décrits.

Ensuite, pour suivre l'évolution de la simulation, il est important de connaître à tout moment la moyenne et l'écart-type des efforts fournis par les agents (le `leffort` pour être précis). En effet, la moyenne permet de voir si l'effort global est élevé ou non. Mais le plus intéressant reste l'écart-type. Si celui-ci est bas, c'est qu'on observe une *corporate culture* comme l'entend l'article. Un écart-type à 0 signifie en effet que tout le monde fournit exactement le même effort. Pour ce faire, nous affichons donc un *plot* pour cette moyenne et un pour l'écart-type, pour évaluer l'évolution de ces valeurs. Un *monitor* pour chacune de ces valeurs est aussi affiché, donnant leur valeur exacte au moment présent.

De même, il est intéressant d'afficher l'évolution des moyenne et écart-type des efforts fournis par chaque type d'agent. On peut imaginer que chaque type d'agent fournit un effort commun différent de celui de ceux des autres agents. Par exemple, avec 200 agents *null effort* et 100 *high effort*, la moyenne se trouve autour de 0.667 et un écart-type, élevé, à environ 0.945. En revanche, comme ces agents ne mettent pas à jour leur effort, chacun de ces types exerce le même effort. On obtient donc les courbes plates de la figure 2, alors que l'écart-type global pourrait signifier de grandes variations.

Il peut être intéressant d'observer le profit dégagé par les agents. C'est pourquoi nous affichons un *plot* donnant son évolution, ainsi qu'un *monitor* donnant la valeur actuelle. Mais il n'est pas réellement nécessaire d'afficher le profit pour chaque agent. En effet, ce profit n'intervient que pour les agents rationnels (Rational et Average Rational), et les Profit Comparator. Cependant, le sujet de l'article

portant sur la « corporate culture » comme un effort commun, cette donnée influençant peu l'effort, il n'est pas nécessaire de la spécifier pour chaque type d'agent.

Cependant, ces plots permettent d'évaluer globalement les agents, suivant leur type ou non. Pour évaluer l'effort d'un agent particulier, il a été rendu possible d'afficher l'effort de chaque agent dans la fenêtre représentant les agents. Comme en NetLogo, il n'est apparemment pas possible d'afficher deux fenêtres avec des agents, il a été décidé d'afficher les agents eux-même sous forme de flèches, et de changer la couleur du patch sur lequel ils se trouvent pour représenter leur effort. Ainsi, on affiche sur la même vue les agents eux-mêmes, repérables grâce à la flèche les représentant, et le dernier effort qu'ils ont fourni, grâce à la couleur du patch sur lequel ils se trouvent. Les figures 2 et 3 de l'article correspondront donc à une seule vue de notre programme, tout comme les deux éléments de la figure 5 de l'article.

A chaque effort correspond une couleur de l'arc-en-ciel, allant du bleu foncé pour l'effort minimal, que nous avons défini par une variable globale `effort-min`, au rouge pour l'effort maximal, que nous avons défini par la variable globale `effort-max`. En respectant l'attribution des couleurs de NetLogo, la formule utilisée est la suivante :

$$\text{pcolor} = \left\lfloor \frac{1}{10} \times (100 - \frac{100 \times \text{leffort}}{\text{effort-max}}) \right\rfloor \times 10 + 15$$

Nous avons d'ailleurs ajouté un *Button* sur l'interface permettant d'exporter une image de la vue actuelle pour les captures d'écran que nous avons eu à réaliser. De même, un bouton permettant d'exporter l'interface sous forme d'image a été implémenté pour les captures d'écran de l'interface entière.

2.2 L'affichage des figures

Pour l'affichage des figures 6 et 7 de l'article, nous avons implémenté plusieurs méthodes pour réaliser les calculs, stocker les résultats, puis les afficher sous forme de *plot*. Nous initialisons tout d'abord l'environnement avec le pourcentage d'agents de chaque type qui nous intéresse. Nous laissons ensuite cette simulation tourner pendant un nombre minimal de ticks, à paramétriser sur l'interface grâce au *Slider* relié à la variable `min-nb-iterations`, tout en gardant dans un tampon la moyenne des efforts des agents. Nous observons alors l'écart-type de ce tampon, et s'il est inférieur à la valeur `ecart-type-max` à régler sur le *Slider* correspondant, on garde la valeur de la dernière moyenne calculée. Sinon on relance la simulation encore une fois, et ce jusqu'à ce que l'écart-type des moyennes d'effort des `min-nb-iterations` dernières itérations soit inférieur à la valeur voulue. Bien entendu, pour éviter une boucle infinie si cela ne converge pas, il faut paramétriser un nombre maximum d'itérations grâce au *Slider* relié à la variable `max-nb-iterations`.

On obtient alors une liste de couples de valeurs (pourcentage de *high effort*, moyenne d'effort obtenue) pour chaque type d'agent recherché. Cette liste est ensuite passée à une méthode (`do-plotting`) qui se charge d'afficher un graphe des valeurs pour chaque liste, donc pour chaque type d'agent, chacune sur une courbe différente.

Pour la figure 9 de l'article, on initialise un environnement dont tous les espaces sont occupés par des *winner imitators*. Ensuite on règle la valeur du bruit à appliquer, et on effectue un *reset-ticks*. Enfin on affiche l'évolution de la moyenne des efforts sur un graphe. Après un certain nombre de ticks donné par la valeur `nb-ticks` associée à un *Slider*, on réitère l'opération avec une autre valeur de bruit.

2.3 Informations notables sur l'implémentation

2.3.1 Initialisation

Conformément à l'article, lors de l'initialisation, l'effort des agents *null effort* est bien évidemment fixé à la valeur de l'effort minimal **effort-min** = 0.0001. De même, les *high effort* et *winner imitator* ont un effort fixé à l'effort maximal **effort-max** = 2.001. Pour les autres agents, on tire aléatoirement pour chaque agent un effort entre ces deux valeurs.

La direction des agents est choisie aléatoirement entre les 4 possibilités données. Toutefois cette direction importe peu, puisqu'elle sera probablement changée au cours de la première étape de l'algorithme, consistant en le mouvement de l'agent. Chacun est placé sur un patch n'accueillant pas déjà un autre agent. Si l'utilisateur rentre plus d'agents que l'environnement peut en accueillir, les agents surnuméraires ne sont pas créés, et un message s'affiche dans le *Command Center* pour le signifier à l'utilisateur.

2.3.2 Implémentation de la mise à jour de l'effort des agents rationnels

Pour les agents rationnels, le calcul de l'effort maximisant son profit, comme indiqué à l'équation 5, comporte le calcul de racines cubiques. Ce calcul étant relativement complexe, nous avons tenté de comparer le temps nécessaire à l'obtention d'un résultat avec d'autres méthodes, en acceptant une certaine marge d'erreur, la plus minime possible.

Pour ce faire, nous avons implémenté deux autres méthodes :

- La méthode de Newton, mais nous n'avons pas constaté une convergence assez rapide.
- L'autre méthode, plus proche de la force brute, consiste à partir du constat que l'on veut maximiser une fonction concave. On peut donc discréteriser l'espace des efforts et trouver la valeur parmi celles-ci qui maximise le profit. On peut partir de l'effort courant, puis essayer de le diminuer et on évalue le profit avec ce nouvel effort. Si on augmente le profit, on réitère l'opération jusqu'à ce que celui-ci recommence à diminuer ou jusqu'à ce qu'on ait atteint la borne inférieure **effort-min**. Au contraire, si le profit diminue, on tente d'augmenter l'effort jusqu'à ce que le profit commence à diminuer ou qu'on arrive à la borne supérieure **effort-max**. On renvoie alors la dernière valeur de l'effort qui n'a pas fait diminuer le profit. Cette méthode peut certes être coûteuse, mais si un équilibre est atteint, comme on démarre toujours de l'effort courant, et qu'il est censé être optimal, on ne fait que vérifier deux autres valeurs d'effort. Donc plus on se rapproche de la valeur de convergence, plus cette méthode rend une valeur rapidement. De plus, plus on diminue le pas entre deux valeurs d'effort à tester, plus on sera proche de la valeur réelle.

Nous sommes donc revenus sur le calcul de l'effort maximisant le profit grâce à la méthode de Cardan. En effet, elle donne la solution exacte de cet effort, quelle que soit l'itération, en un temps très faible.

3 Expérimentations et reproduction des résultats

3.1 Émergence d'une corporate culture

Il existe des cas où tous les agents de la simulation finissent par fournir le même effort, au bout d'un temps assez variable. Par exemple, il n'est pas surprenant que dans un environnement uniquement constitué d'agents *high effort*, aucun d'eux ne modifiant son effort, un effort unique identique et maximal est exercé par tous les agents.

Cela dit, pour d'autres configurations, c'est moins évident. Lorsque la population est uniquement constituée d'agents rationnels, les agents modifient très rapidement leur effort pour arriver à l'équilibre

de Nash du système : 0.92101. C'est ce qu'on peut voir sur la figure 3. En environ une trentaine d'itérations, tous les agents fournissent le même effort, ce qu'on peut vérifier avec l'écart-type qui est nul. Cela s'explique par le fait que dans cette configuration, les agents cherchent tous à maximiser leur profit. Donc si deux agents se font face et ont un effort de 0.92101, aucun d'eux n'a intérêt à modifier son effort, car il réduirait son profit.

Dans le cas d'une population constituée uniquement d'agents *shrinking effort*, l'effort global va diminuer, montrant une courbe exponentielle décroissante, pour se stabiliser à un effort nul. Cela s'explique très simplement par le fait que ces agents fournissent un effort deux fois moindre que celui de leur dernier partenaire. Comme aucun d'eux ne fixe son effort, même si l'un des deux partenaires fournit un effort élevé, l'autre ne le conservera qu'à moitié, et en plus il ne transmettra que la moitié du sien. Le côté exponentiel décroissant vient de la division par 2. En effet, à chaque itération, l'effort global de tous les agents effectuant un « jeu » est divisé par deux. Il ne suffit donc que d'un nombre restreint d'itérations pour observer ce résultat, comme on peut le voir sur la figure 4.

Dans le cas d'une population d'agents *replicator* avec un seul agent à effort fixe, comme un agent *null effort* ou *high effort*, on observe une convergence plus ou moins lente selon la densité d'agents dans l'environnement, vers l'effort exercé par l'agent à effort fixe. Cela est dû au fait que les *replicator* collaborant avec cet agent vont récupérer cet effort, mais ne pas transmettre le leur puisque leur partenaire a fixé le sien. Donc à chaque fois que l'agent à effort fixe interagit avec un *replicator*, il diminue le nombre d'agents exerçant un effort différent du sien. Le temps de convergence correspond donc ici au temps que l'agent à effort fixe prend pour transmettre son effort à tous les agents (directement ou non). C'est très aléatoire, mais la probabilité que cet agent à effort fixe rencontre un agent qui n'a pas encore le même effort que lui étant non-nulle, d'après le lemme de Borel-Cantelli, cela arrivera avec probabilité 1. On peut observer cette convergence en un temps fini sur les figures 5 et 7, où l'on voit respectivement la convergence progressive des efforts des *replicator* vers 0.0001 et 2.001.

Si l'environnement ne comporte que des agents *replicator* à l'exception d'un agent rationnel, le système converge vers un effort correspondant au même équilibre de Nash que s'il y avait uniquement des agents rationnels : 0.92101. En effet, L'agent rationnel va progressivement être copié par les agents qu'il rencontre, puis ceux-ci vont à nouveau le rencontrer, et l'agent rationnel va se rapprocher de plus en plus de l'équilibre de Nash cité précédemment, comme s'il avait en face de lui un agent qui suivait la même stratégie que lui. Et c'est en quelque sorte le cas, puisqu'à chaque fois qu'il s'ajuste pour maximiser son profit, il est copié par ceux qu'il rencontre. La figure 6 le prouve.

3.2 Absence de convergence

Cependant, il se peut qu'aucune convergence ne puisse se produire, et on n'observe pas de « corporate culture », au sens de l'article, se former. Le cas le plus simple est celui où l'environnement ne comporte que des agents *replicator*. Puisqu'ils ne font que s'échanger leurs efforts, aucun effort ne change réellement, ils ne font que se transmettre d'un agent à un autre. La moyenne globale et l'écart-type n'évoluant pas, leurs courbes sont donc parfaitement plates, comme le prouve la figure 8.

De même, si l'on place des agents *replicator* dans une population constituée de *null* et *high effort*, l'effort des premiers va osciller autour de la moyenne des efforts des agents des deux autres types. En effet, selon leur rencontre, ils vont soit imiter un type, soit l'autre, et donc basculer incessamment. C'est ce qu'on observe à la figure 9.

Cela peut être observé de la même manière avec les agents de type *averager*, *rational* et *average rational*.

Nous avons aussi reproduit la figure 5 de l'article, montrant qu'une population de *effort comparator* et de *high effort* ne pouvait mener à une convergence. En effet, là où les agents *high effort* sont le plus concentrés, l'effort autour d'eux est plus élevé qu'ailleurs. C'est tout à fait logique, puisque c'est dans

ces régions que l'autre type d'agent a le plus de probabilités d'interagir avec un *high effort*, ou avec quelqu'un qui l'a fait. On observe donc des îlots se former autour des grandes densités de ce type agent. La figure 10 montre cet état de faits.

On peut associer cette image avec la présence des chefs d'équipe dans les entreprises. Autour de ceux-ci l'effort reste élevé, pour ne pas se faire réprimander. En revanche lorsque l'agent est éloigné, il a tendance à penser qu'il est loin des problèmes et fournir moins d'effort.

Cette constatation peut être faite avec les agents *profit comparator* pour les mêmes raisons. On peut le voir sur la figure 11.

3.3 L'impact des agents à effort élevé

Comme expliqué dans la section 2.2, nous avons lancé la simulation pour chaque type d'agent avec le pourcentage voulu d'agents *high effort* dans l'environnement jusqu'à ce que l'écart-type des moyennes observées sur les dernières simulations soit assez bas, tout en ne dépassant pas un certain palier. Les résultats sont ensuite stockés sous forme de listes, puis affichés dans le plot prévu.

Dans nos simulations, nous avons pris tout d'abord un espace de 30 patches de côté, puis de 15 patches de côté sans changement de résultat, mais avec un gain de temps d'exécution notable. Nous avons la plupart du temps utilisé un nombre minimal d'itérations **min-nb-iterations** de 1000, dans l'espoir d'avoir assez de valeurs à comparer pour évaluer si un échantillon varie ou non. Le nombre maximal d'itérations **max-nb-iterations** a été réglé à 20000 car nous nous sommes rendus compte que lorsqu'une convergence était à attendre, elle se produisait souvent avant ce nombre d'itérations. En particulier, lorsque le pourcentage d'agents *high effort* est faible, la courbe de *averager* prend un certain temps à atteindre la valeur maximale. Quant à l'écart-type **ecart-type-max**, qui est un critère d'arrêt de la simulation, nous avons choisi de le paramétriser assez bas, à 0.001.

3.3.1 Reproduction de la figure 6 de l'article

La figure 12 donne le résultat obtenu lorsque nous tentons de reproduire la figure 6 de l'article, montrant l'évolution de la moyenne d'effort fourni par les agents dans une population où la proportion de *high effort* augmente. Quel que soit le type d'agent, l'effort global augmente avec ce pourcentage. Il n'y a qu'une exception, ce sont les agents *winner imitator* qui stagnent, puisque ces derniers commencent déjà à l'effort maximal, et ne sont donc pas plus influencés par les *high effort* que par les autres agents du même type qu'eux. Cela s'explique principalement par le fait que la présence de ces agents à effort élevé tire naturellement vers le haut la moyenne globale, même si les autres agents restent à 0.

Ce dernier constat est le plus flagrant lorsqu'on regarde la courbe associée aux agents *null effort*. Ils ne mettent jamais à jour leur propre effort, et pourtant l'effort global augmente. C'est uniquement dû à la présence des *high effort*. La valeur ainsi affichée à chaque point n'est autre que la proportion de ces derniers multipliée par leur effort, si l'on considère que l'effort fourni par les *null effort* est négligeable.

Pour les agents *shrinking effort*, on peut observer que leurs valeurs sont légèrement au dessus de celles des *null effort*. C'est principalement dû au fait que plus il y a d'agents *high effort*, plus ils ont de chance d'en rencontrer et donc d'exercer un effort moyen. Leur effort ne va donc plus tendre vers 0 mais remonter de temps en temps lorsqu'ils rencontrent un agent à effort élevé.

Pour les agents *profit comparator* et *effort comparator*, la moyenne d'effort augmente très rapidement dès l'introduction d'agents *high effort*. Dans ces deux cas, la présence de ces derniers force les agents de type *comparator* à augmenter leur effort car le profit, respectivement l'effort, des agents à effort élevé sera très souvent supérieur au leur. De plus, comme ceux qui augmentent leur effort vont influencer ceux du même type qui vont à leur tour augmenter leur effort. Mais s'il n'y a pas assez de *high effort*, aucune convergence ne se produit, les agents proches de ceux-ci exhortant un effort élevé alors que les autres ne fourniront qu'un effort beaucoup plus faible, comme nous l'avons expliqué dans la section 3.2.

Les agents *replicator*, comme nous l'avons décrit dans la section 3.1, ces agents vont imiter les uns après les autres les agents *high effort*, qui eux ne changent pas leur effort. Ainsi, la proportion de ces derniers n'influence pas le résultat, mais uniquement le temps nécessaire à son obtention.

Les agents *winner imitator* commençant avec un effort maximal identique à celui des *high effort*, le profit et l'effort de tout le monde est identique. Donc même s'ils gagnaient contre les agents *high effort*, ils ne feraient que copier un effort qui est similaire au leur. C'est pourquoi leur courbe est plate, fixée à l'effort maximal.

Dans le cas des agents de type *averager*, dès qu'un seul agent de type *high effort* est introduit dans la population, l'ensemble va tendre de façon logarithmique vers l'effort maximal si on lui donne assez de temps. C'est pourquoi le graphe montre que sans agent *high effort* un équilibre est trouvé à la moyenne des efforts de départ. Dans ce cas, on peut observer que la moyenne global d'effort ne change pas, mais l'écart-type va baisser rapidement pour devenir nul. En revanche, quelle que soit la proportion non nulle d'agents à effort maximal, la moyenne globale va tendre vers cet effort. Comme pour les agents de type *replicator*, cette proportion n'influencera que la vitesse de convergence.

Si l'on se penche plus précisément sur les valeurs de ce graphe, on remarque que pour 5.6% d'agents *high effort*, l'article suppose un effort moyen pour les *effort comparator* et *profit comparator* aux alentours de 0.55. Nous avons fait tourner notre simulation avec différents paramètres, et nous nous approchons en moyenne les 1.40. Cela est probablement dû au fait qu'à ce pourcentage de *high effort*, aucune convergence n'est observée, comme nous l'avons expliqué dans la section 3.2. La valeur retournée dépend alors de la répartition des agents *high effort*. Plus ces derniers sont uniformément répartis, plus l'effort général sera bas. D'ailleurs, lors de certains lancers, nos valeurs ont approché les valeurs attendues par l'article, comme le montre la figure 13. À 5.6% de *high effort* dans une population de *effort comparator*, nous avons trouvé un effort moyen d'environ 0.53.

Au contraire, à 33.3% de *high effort*, leur moyenne d'effort s'approche plutôt de 1.95, alors que la nôtre est plus proche de 1.89. Cette fois, la raison peut probablement se trouver dans la précision sur les flottants qui n'est pas bien gérée par NetLogo.

Le reste des valeurs reste vraiment très proche des valeurs attendues et affichées dans l'article. De plus, il est important de noter que la forme globale des courbes est très similaire entre les résultats de l'article et les nôtres. Ces différences sont probablement dues à des précisions sur les calculs de flottants, ou à une précision différente sur les pourcentages puisqu'on doit prendre un nombre entier de chaque type d'agents, et il est donc impossible de prendre précisément par exemple 0.6%, ou 5.6% d'un type d'agents sans réaliser les calculs sur un nombre immense d'agents.

3.3.2 Reproduction de la figure 7 de l'article

Dans la figure 7 de l'article, on étudie l'évolution de l'effort moyen d'une population constituée d'agents rationnels à laquelle on ajoute des agents de type *high effort*. Comme expliqué dans la section 3.1, sans agent *high effort*, la simulation tend vers l'équilibre de Nash du système. Pour les mêmes raisons, lorsqu'on ajoute ce type d'agents, l'équilibre de Nash du système est atteint lorsqu'on s'éloigne des agents à effort élevé. Cependant, près de ceux-ci, les agents rationnels ont tendance à diminuer leur effort pour maximiser leur profit. L'effort global augmente néanmoins, mais cela est dû à la présence de ces agents *high effort* qui tire la moyenne vers le haut, ainsi que des agents rationnels qui restent à l'équilibre de Nash.

Ce résultat peut être observé sur la figure 14.

3.4 L'impact du bruit

Le bruit impacte la perception de l'effort du partenaire par les agents. Dans une population constituée uniquement d'agents de type *winner imitator*, ces agents vont pouvoir croire que leur partenaire a eu un meilleur profit en abaissant son effort, et donc l'imiter. Cela se fait itérativement jusqu'à ce

que tous les agents exercent un effort virtuellement nul. Cela peut se voir sur la figure 8 de l'article, et la figure 15 de ce rapport. On observe que l'effort global des agents diminue de façon plus ou moins simultanée vers une gamme d'effort inférieur jusqu'à l'effort minimal. Les images de l'article peuvent donner l'impression que leur population est plus homogène que la nôtre. Mais il est fort probable que leur plage de couleur soit moins fine que la nôtre. D'ailleurs, lorsque la diminution est plus lente, sur les dernières images, on voit aussi de notre côté que l'effort de tous les agents diminue presque de concert.

Ce résultat peut être reproduit avec d'autres populations homogènes telles que les *replicator* ou les *averager*. Pour ce dernier type d'agent, bien qu'il varie beaucoup avant de converger, il finit toujours par converger vers un effort minimal. Ces deux derniers cas s'expliquent par le fait que le bruit a plus d'impact sur un effort élevé que sur un bas. En effet, même 1% d'un effort de valeur 2 est plus élevé que 1% d'un effort de valeur 0.0001. Donc une fois arrivé à une valeur aussi basse, remonter grâce au bruit est compliqué, alors que descendre à partir d'une valeur élevée est très aisée.

En revanche, quelle que soit la valeur du bruit, sur une population uniquement composée d'agents rationnels, celui-ci n'a que peu d'effet. Tous les agents convergent vers l'équilibre de Nash malgré quelques fluctuations dont l'ampleur dépend de l'intensité du bruit. La figure 16.

Quant à l'effet de la valeur du bruit appliquée, il est assez intéressant. Pour une valeur faible de 1%, l'effort moyen diminue déjà assez rapidement pour tendre vers l'effort minimal. Ensuite, La pente de dégradation initiale augmente vraiment rapidement avec la valeur du bruit. Si on compare notre figure 17 avec la figure 9 de l'article, on constate que la valeur du bruit moyen est d'environ 5%, et non pas 25% comme on pourrait naïvement le penser. C'est probablement dû au fait que le bruit s'applique sur les deux agents qui collaborent, et non un seul. Ceci expliquerait l'effet de la valeur du bruit qui paraît quadratique.

Encore une fois, on retrouve la même forme globale des courbes.

3.5 Suggestions d'améliorations du modèle

Nous avons plusieurs suggestions d'amélioration de ce système pour le rendre plus réaliste.

La première consiste à faire évoluer les agents, donc à ne pas considérer leur type comme fixe dès le départ. On pourrait imaginer qu'ils changeraient de type par le biais d'une récompense. S'ils sont récompensés, ils auraient tendance à augmenter leur effort, sinon à le diminuer. Par exemple, cette récompense pourrait correspondre au rapport $\frac{\text{effort}}{\text{profit}}$. Le changement de type pourrait alors se produire vers un type aléatoire, ou vers un type particulier.

Une seconde proposition consiste à permettre à tous les types d'agents, dès qu'ils interagissent avec quelqu'un, de ne pas prendre en compte uniquement l'effort de l'agent avec qui ils collaborent directement, mais celui de tout le voisinage.

La dernière, de nature moins éthique mais probablement plus réaliste, consiste à licencier à chaque itération une certaine proportion de la population la moins efficiente, puis de réembaucher le même nombre d'agents. La détection de l'effort fourni par les agents pourra être soumis à un bruit pour conserver un certain réalisme.

Quant à l'embauche, elle peut se faire soit de façon aléatoire, soit de façon à garder plus ou moins la même répartition des agents restants, pour essayer de conserver cette « corporate culture » recherchée. Dans ce deuxième cas, il peut être intéressant d'avoir un certain bruit sur la détection du type d'agent qu'on embauche. On essaie alors d'embaucher les agents dans les mêmes proportions, mais des fois on se trompe et on prend un agent aléatoirement.

De plus, on peut imaginer un système de départ à la retraite, démission, et période d'essai dans ce contexte. Pour la période d'essai et la retraite, il suffit de mettre des bornes pour le licenciement. À moins d'un certain nombre de collaborations on ne peut pas être licencié, et avec un nombre trop

important de collaborations, on quitte l'entreprise pour partir à la retraite. Pour les démissions, il suffit de prendre une probabilité que l'agent démissionne à chaque itération et de l'appliquer.

3.6 Implémentation d'une amélioration

3.6.1 Implémentation

Nous avons tenté d'implémenter ces variantes, mais nous nous sommes concentrés sur la dernière, qui nous semblait très intéressante, complète, et assez réaliste. Nous avons donc donné la possibilité à l'utilisateur de paramétriser toutes les variables citées dans la description de cette amélioration. On peut voir les moyens de régler ces variables sur la figure 18 qui est une copie d'écran de l'interface liée à ces améliorations. Notons qu'il a été nécessaire d'ajouter des *Monitor* pour observer les différentes populations d'agents, puisqu'elles sont maintenant variables au cours du temps.

Pour simuler un renouvellement des effectifs, les licenciements se font seulement si l'écart-type des efforts des agents dépasse un certain seuil **ecart-type-renouvellement**, paramétrable via l'interface. A partir de là, nous identifions les agents fournissant le moins d'effort. Si l'on dénombre un ensemble d'agents dépassant un certain pourcentage de l'ensemble global, on ne retient que le bon nombre d'agents à licencier.

Suite à cette étape vient un temps d'embauche, où l'on ajoute le même nombre d'agents que le nombre de licenciés, où, au choix, l'on rajoute des agents de type aléatoire, ou de façon à respecter la proportion des agents toujours présent dans l'environnement.

Un bruit peut être ajouté, que ce soit lors des licenciements ou des embauches.

Étant donnée notre implémentation, il est à noter que si l'écart-type global est nul, aucun licenciement ne sera fait. Cela veut dire que l'entreprise a atteint son objectif : une « corporate culture ». Ainsi, même si tous les agents sont de type *null effort*, aucun changement ne sera fait.

3.6.2 Résultats

Quelques configurations émergent, selon l'état initial de la simulation, et les valeurs des paramètres, notamment concernant les bruits.

Lorsque l'intensité du bruit est faible, et que l'embauche se fait de façon à garder les mêmes proportions, les simulations tendent vers un ensemble d'agents majoritairement de type *replicator*, *high effort* ou *averager*. Lorsque initialement il y a une majorité d'agents de type *null effort* ou *shrinking effort*, le système converge vers un ensemble d'agents principalement de type *high effort*, comme le montrent les figures 19 et 20.

En revanche si l'on rajoute ne serait-ce qu'un faible pourcentage d'agents de type *averager* ou *replicator* au début de nos simulations, puisque l'on cherche à garder les proportions à chaque embauche, nous verrons apparaître ce type d'agent jusqu'à la convergence, puisque ceux qui ne sont pas licenciés tendent à se comporter comme des agents de type *high effort*, comme le montre les figures 21 et 22.

Tous dépend des proportions initiales. Il faut par exemple assez de type *replicator* pour qu'ils puissent persister tout au long de la simulation. On peut voir une différence entre la figure précédente, et la figure 23

Lorsque l'on débute avec un ensemble d'agents majoritairement rationnels, ceux-ci ont tendance à persister, en convergeant vers l'équilibre de Nash. En effet, l'écart-type tend rapidement vers zero. La figure 24 illustre ce résultat.

Lorsque l'on rajoute du bruit à l'embauche, le taux d'agents de type *high effort* reste élevé, mais viennent s'ajouter des agents de type *averager* et *replicator*, de façon significative. Durant la simulation, des *averager* et *replicator* ont plus de chance d'être embauchés, et ont donc le temps de s'adapter et d'agir comme les *high effort*, d'où leur persistance. La figure 25 montre cela.

D'après les résultats obtenus à la figure 26, on ne retrouve pas la persistance des types que l'on pouvait observer lors d'une simulation ne comportant que des agents de type *rational* et *average rational*. Ceux-ci étaient dûs au faible écart-type, qui tendait vers zéro grâce à la conservation des proportions. Or ce point de convergence est instable, il suffit d'infilttrer dans l'environnement un faible nombre d'agents de type *high effort* par exemple, pour faire basculer la convergence, auquel cas, on finit par avoir une majorité de type *high effort*.

Les figures 19 à 30 montrent d'autres convergences qui ont pu être observées, mais ne méritent pas que l'on s'étende sur elles, leur mécanisme étant trivial.

Conclusion

La question de la « corporate culture » est très importante pour le milieu économique actuel. C'est le gage d'un effort commun, dans un but commun. Mais c'est surtout une image de marque qui se dégage à la fois pour les consommateurs et pour les éventuels futurs employés. Les meilleurs éléments pourraient être soit attirés, soit repoussés par une culture d'entreprise trop différente de la leur. Cet article tente de montrer les configurations de « personnalités » présentes dans une entreprise aboutissant à la formation d'une culture d'entreprise.

Il étudie aussi l'effet de variables telles que le *bruit* de détection de l'effort, qui fait majoritairement baisser l'effort global. Ce résultat incite les entreprises à encourager la communication et une collaboration transparente entre toutes les parties d'une entreprise dans le but de réduire un maximum l'intensité de ce bruit. Ainsi, les chances de créer une dynamique de groupe se voit largement augmentées.

L'article donne alors un certain nombre de configurations pour la population initiale sensées mener à une culture d'entreprise. De plus, il est spécifié si celle-ci se fait vers un effort global élevé ou faible. Même, lorsqu'aucun équilibre n'est trouvé, l'article donne les raisons de cette absence de convergence, pour éventuellement pouvoir y remédier. Par exemple, la répartition en îlots d'effort élevé autour des grandes densité d'agent *high effort* dans une population d'agents *effort comparator* ou *profit comparator* pourrait inciter les entreprises à réorganiser l'emplacement de leurs meilleurs éléments. Le but étant qu'ils influencent un maximum de personne, au lieu d'être isolés, et donc n'avoir que peu d'effet sur la population générale.

Notre amélioration, qui tente de simuler le « *Turn Over* » dans les entreprises montre aussi que ce ne sont pas forcément les agents *high effort* qui donnent une *corporate culture* avec un effort élevé. Ce sont ceux qui tentent de les imiter, et donc les plus motivés, qui sont le vrai moteur d'une entreprise.

A Annexes

Table des figures

1	Portion de l'interface pour le réglage des paramètres	17
2	Moyennes et Ecart-types affichés pour 200 <i>null effort</i> et 100 <i>high effort</i>	18
3	Résultats obtenus avec une population uniquement constituée d'agents rationnels	18
4	Résultats obtenus avec une population uniquement constituée d'agents <i>shrinking effort</i>	19
5	Résultats obtenus avec une population constituée d'agents <i>replicator</i> avec un seul agent <i>null effort</i>	19
6	Résultats obtenus avec une population constituée d'agents <i>replicator</i> avec un seul agent <i>rational</i>	20
7	Résultats obtenus avec une population constituée d'agents <i>replicator</i> avec un seul agent <i>high effort</i>	20
8	Absence de convergence avec uniquement des agents <i>replicator</i>	21
9	Absence de convergence avec des agents <i>replicator</i> en présence de deux autres types d'agents d'efforts contraires	21
10	Îlots d'effort plus élevé autour d'agents <i>high effort</i> dans une population de <i>effort comparator</i>	22
11	Îlots d'effort plus élevé autour d'agents <i>high effort</i> dans une population de <i>profit comparator</i>	22
12	Effet de la présence d'agents <i>high effort</i> sur une population d'agents à la rationalité limitée	23
13	Effet de la présence d'agents <i>high effort</i> sur une population d'agents à la rationalité limitée	23
14	Effet de la présence d'agents <i>high effort</i> sur une population d'agents rationnels	24
15	Effet du bruit sur une population homogène d'agents <i>winner imitator</i>	24
16	Evolution de l'effort global d'une population de <i>rational</i> s avec 50% de bruit	25
17	Evolution de l'effort global d'une population de <i>winner imitator</i> suivant différentes valeurs de bruit	25
18	Capture d'écran de la portion de l'interface permettant le paramétrage des améliorations	26
19	Renouvellement d'effectif avec 1% de bruit. Population homogène de <i>shrinking effort</i> , convergence vers un ensemble d'agents de type <i>high effort</i>	26
20	Renouvellement d'effectif avec 1% de bruit. Population initiale de <i>null effort</i> et <i>shrinking effort</i> , convergence vers un ensemble d'agents de type <i>high effort</i>	27
21	Renouvellement d'effectif avec 1% de bruit. Population initiale hétérogène de <i>null effort</i> et <i>shrinking effort</i> , et peu de <i>replicator</i> et <i>averager</i> . Les type <i>replicator</i> et <i>averager</i> persistent.	28
22	Renouvellement d'effectif. Convergence vers le type <i>replicator</i>	29
23	Renouvellement d'effectif avec 1% de bruit. Population initiale de <i>null effort</i> et <i>shrinking effort</i> et peu d' <i>averager</i> (les <i>replicator</i> ne sont pas assez nombreux pour se démarquer). Les type <i>averager</i> persistent.	30
24	Renouvellement d'effectif avec 1% de bruit. Population initiale de <i>rational</i> et <i>average rational</i> . Ces agents persistent, en convergeant leurs efforts vers l'équilibre de Nash.	31
25	Renouvellement d'effectif, avec 18% de bruit. Population initiale de <i>null effort</i> et <i>shrinking effort</i> . Ces agents persistent, en convergent leurs efforts vers l'équilibre de Nash.	32
26	Renouvellement d'effectif avec 18% de bruit. Instabilité de l'équilibre de Nash.	33
27	Renouvellement d'effectif avec 1% de bruit. Covergence vers le type <i>replicator</i>	34
28	Renouvellement d'effectif. Stabilité de l'équilibre de Nash.	35
29	Renouvellement d'effectif. Stabilité du type <i>averager</i>	36

30	Renouvellement d'effectif. Population initiale de <i>null effort</i> et d'un <i>shrinking effort</i> . Émergence des agents rationnels lorsqu'il y a du bruit au licenciement et à l'embauche.	37
31	Renouvellement d'effectif. Population initiale de <i>shrinking effort</i> et d'un <i>null effort</i> . Émergence des agents rationnels lorsqu'il y a du bruit au licenciement et à l'embauche.	38

A.1 Figures

Paramètres

Nombre d'agents de chaque type

null effort

nb-agents-0

200

high effort

nb-agents-5

100

shrinking effort

nb-agents-1

0

average rational

nb-agents-6

0

replicator

nb-agents-2

0

winner imitator

nb-agents-7

0

rational

nb-agents-3

0

effort comparator

nb-agents-8

0

profit comparator

nb-agents-4

0

averager

nb-agents-9

0

Divers

display-effort

On
 Off

allowed-step

1

Bruit

noise?

On
 Off

Intensité du bruit :

noise-value

1 %

FIGURE 1 – Portion de l'interface pour le réglage des paramètres

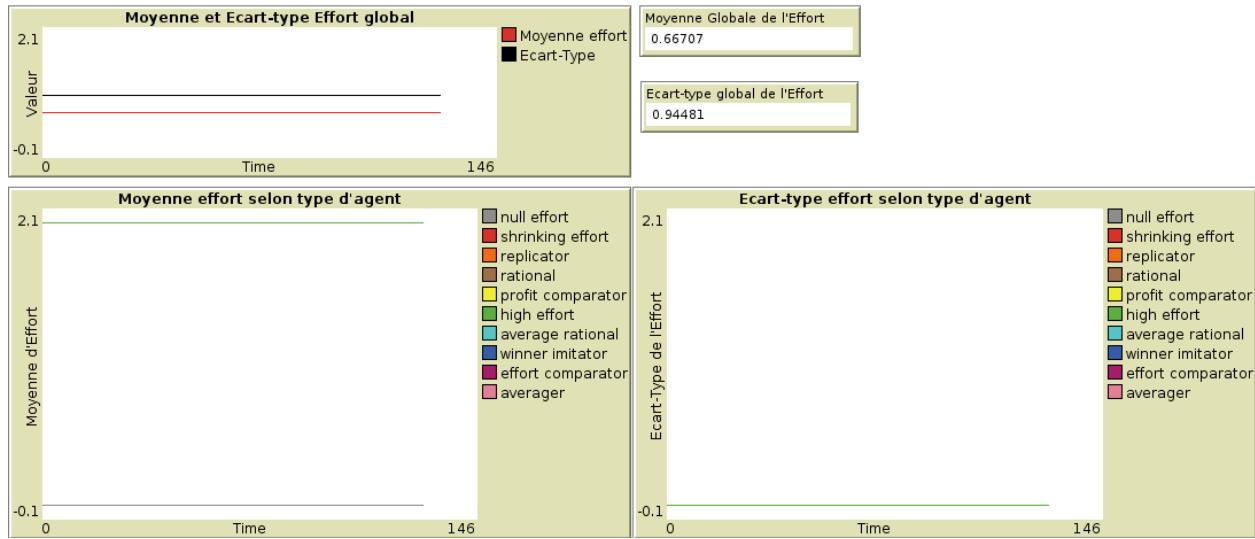
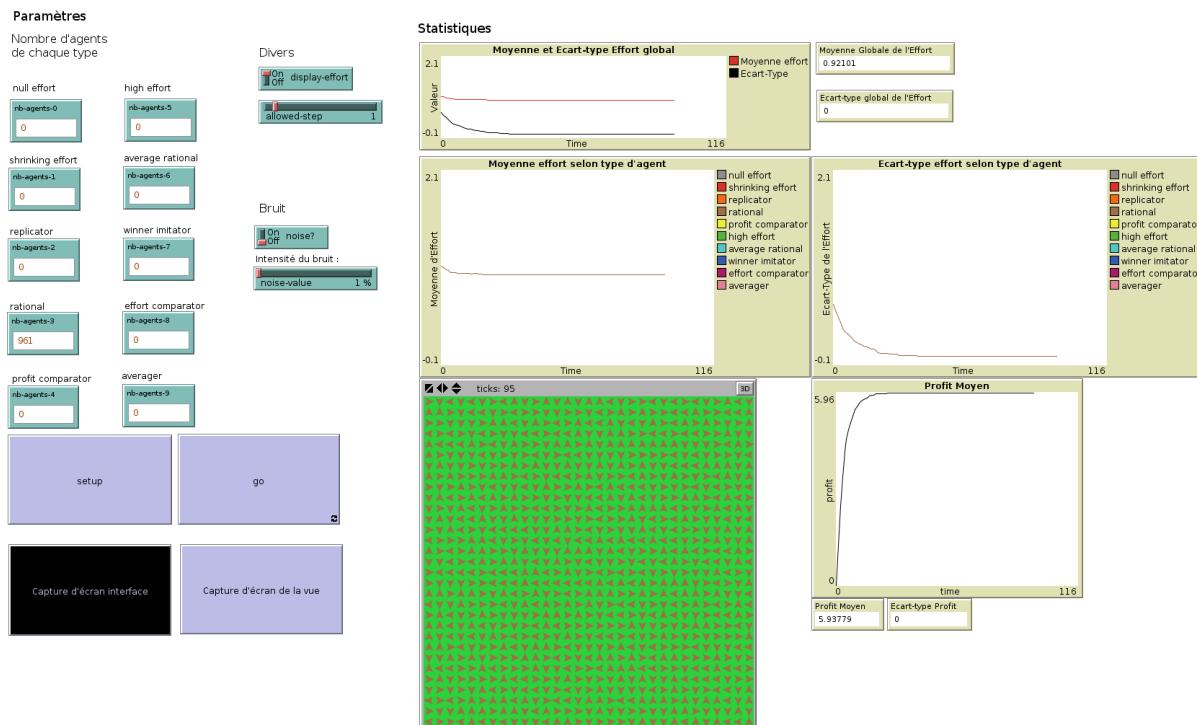
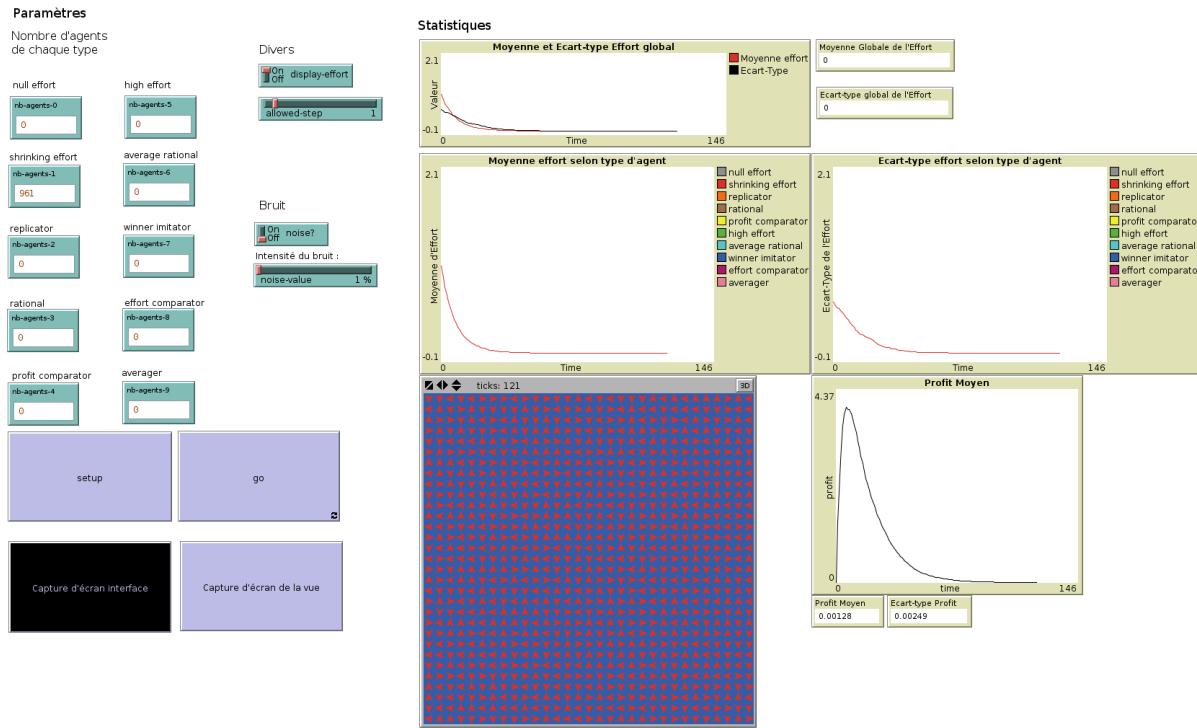
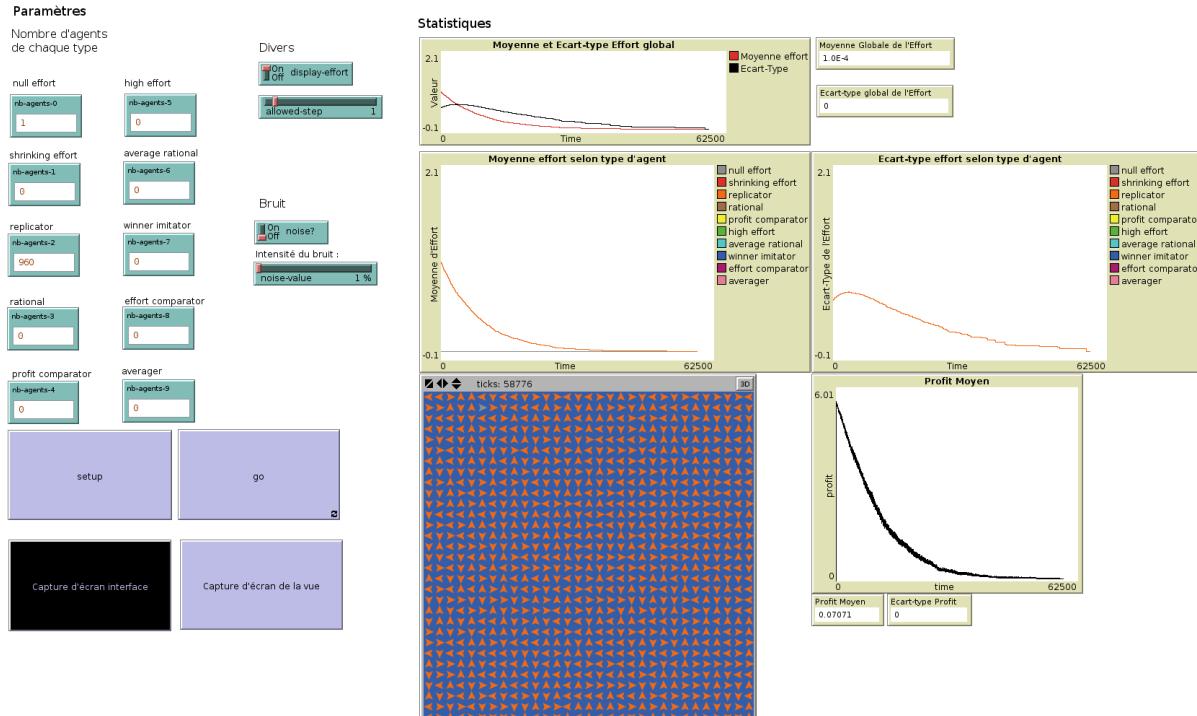
FIGURE 2 – Moyennes et Ecart-types affichés pour 200 *null effort* et 100 *high effort*

FIGURE 3 – Résultats obtenus avec une population uniquement constituée d'agents rationnels

FIGURE 4 – Résultats obtenus avec une population uniquement constituée d'agents *shrinking effort*FIGURE 5 – Résultats obtenus avec une population constituée d'agents *replicator* avec un seul agent *null effort*

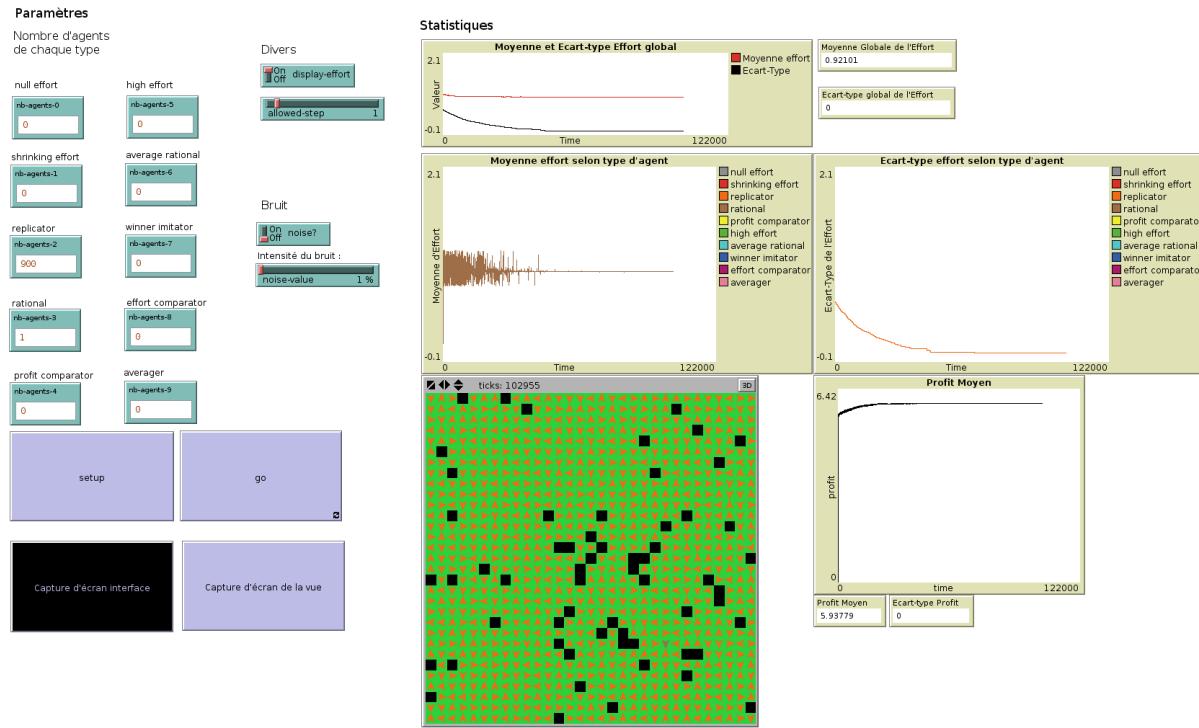


FIGURE 6 – Résultats obtenus avec une population constituée d'agents *replicator* avec un seul agent *rational*

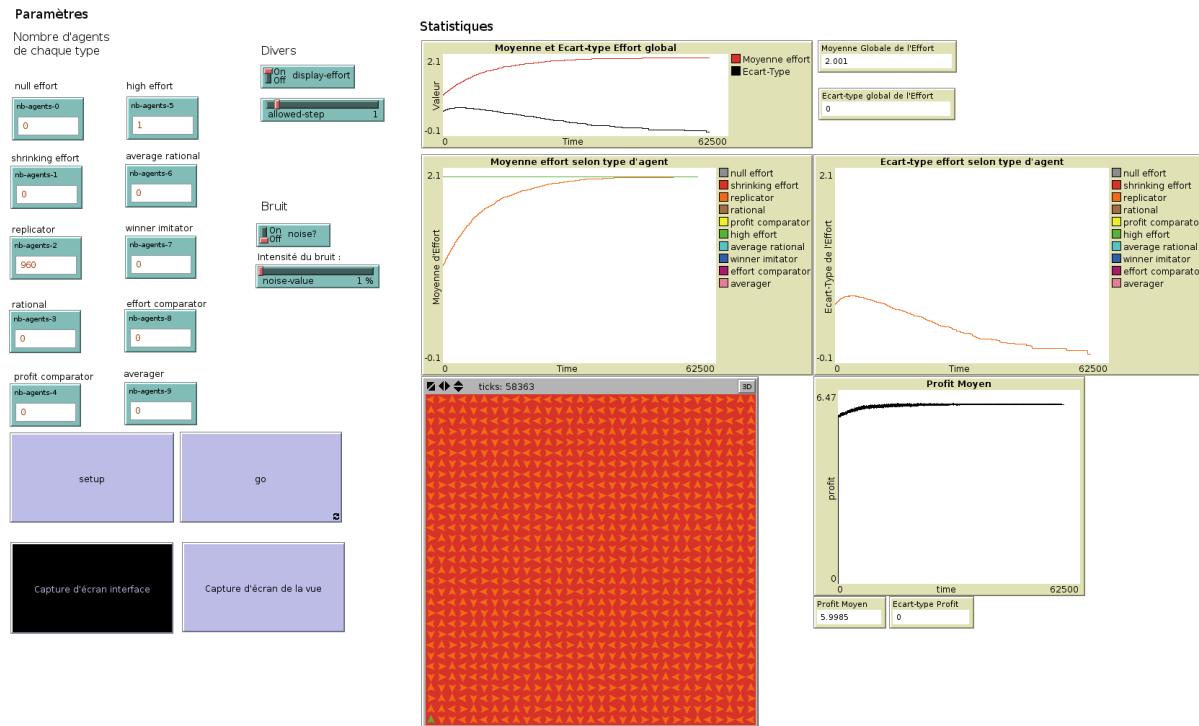
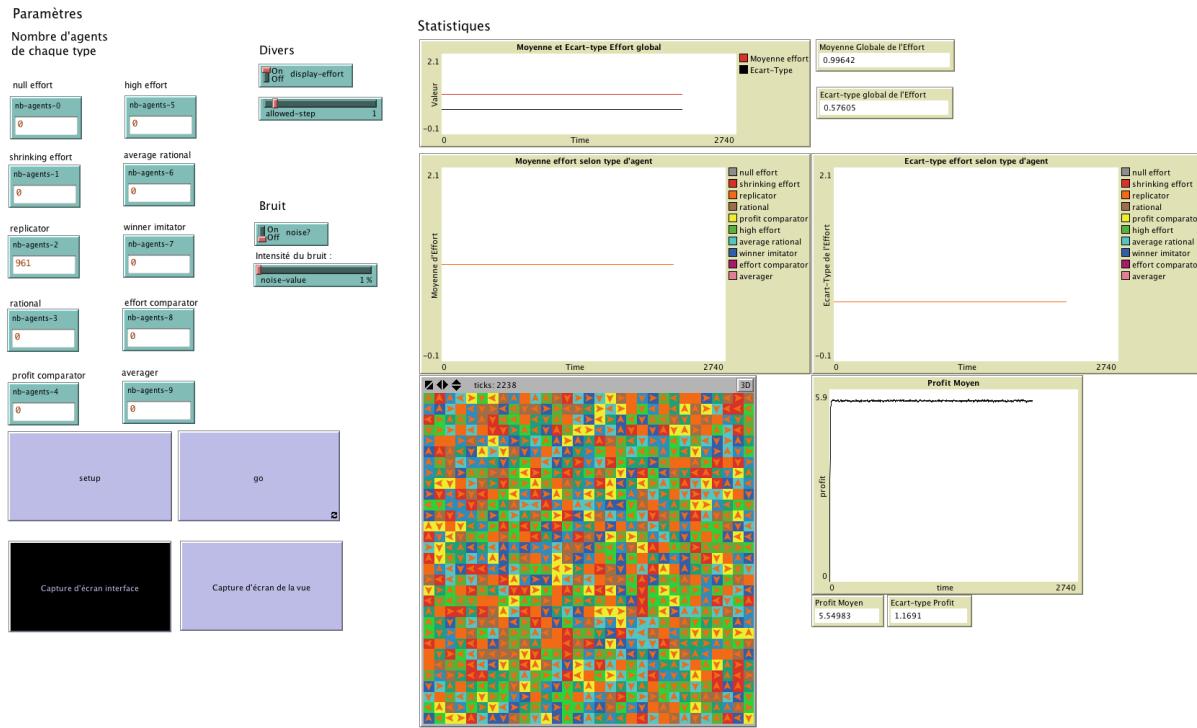
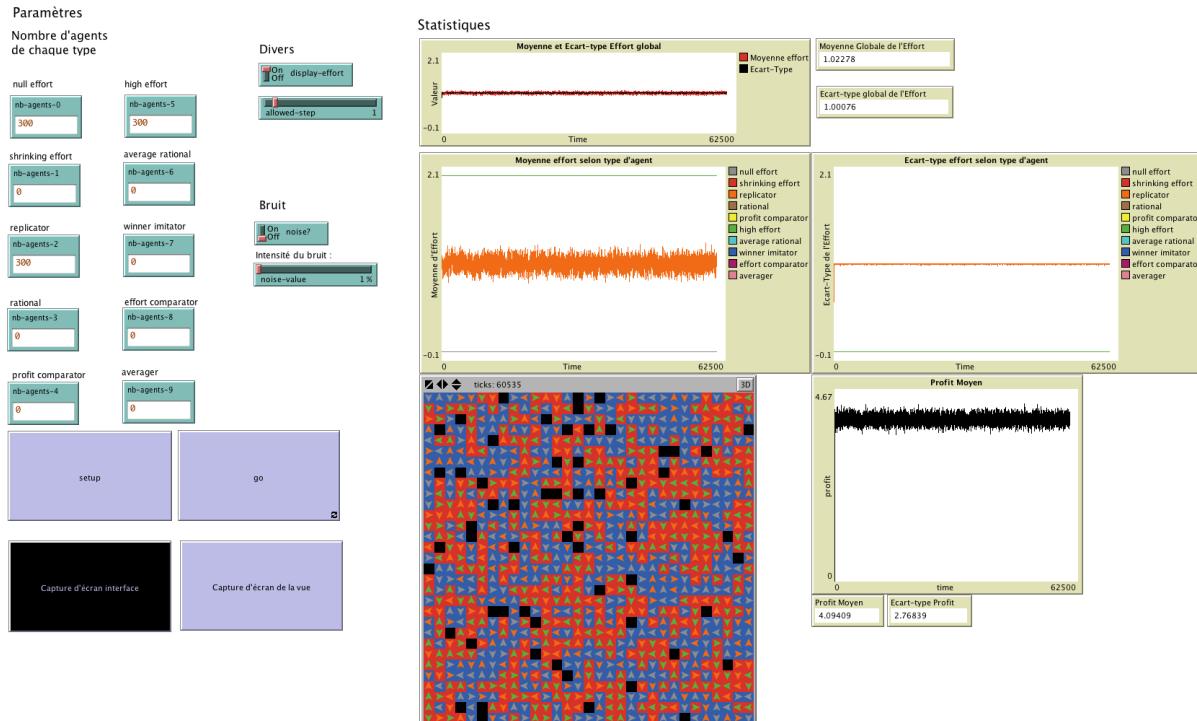


FIGURE 7 – Résultats obtenus avec une population constituée d'agents *replicator* avec un seul agent *high effort*

FIGURE 8 – Absence de convergence avec uniquement des agents *replicator*FIGURE 9 – Absence de convergence avec des agents *replicator* en présence de deux autres types d'agents d'efforts contraires

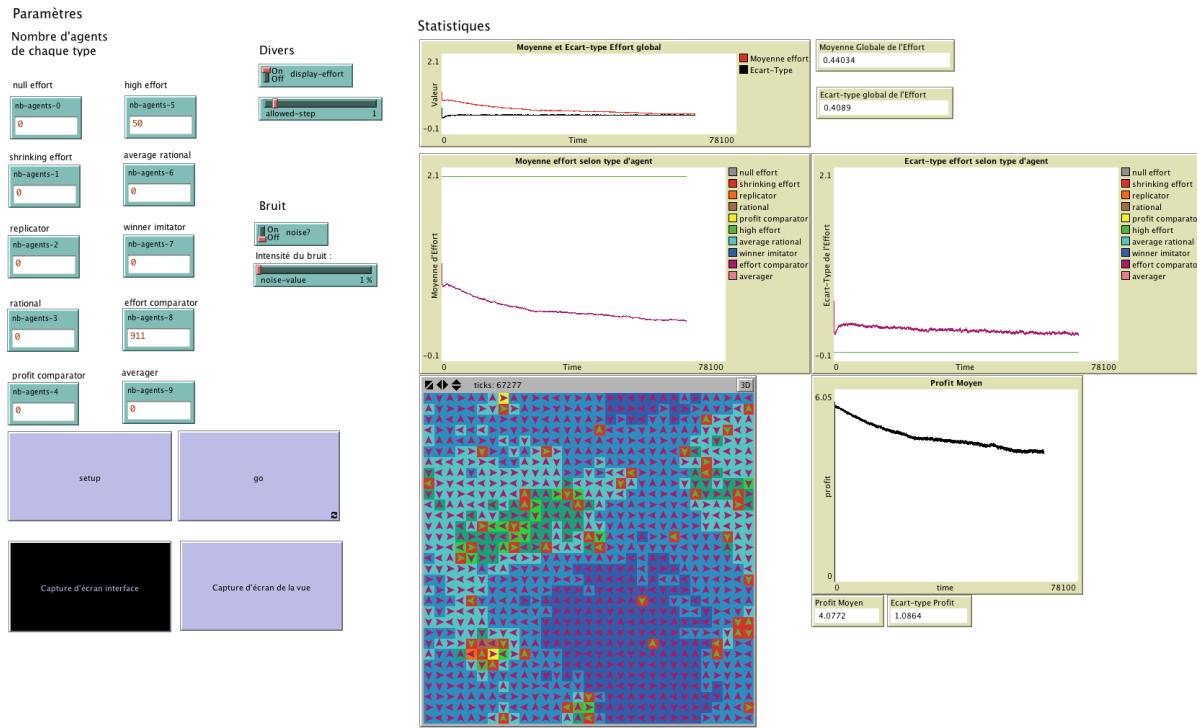


FIGURE 10 – Îlots d'effort plus élevé autour d'agents *high effort* dans une population de *effort comparator*

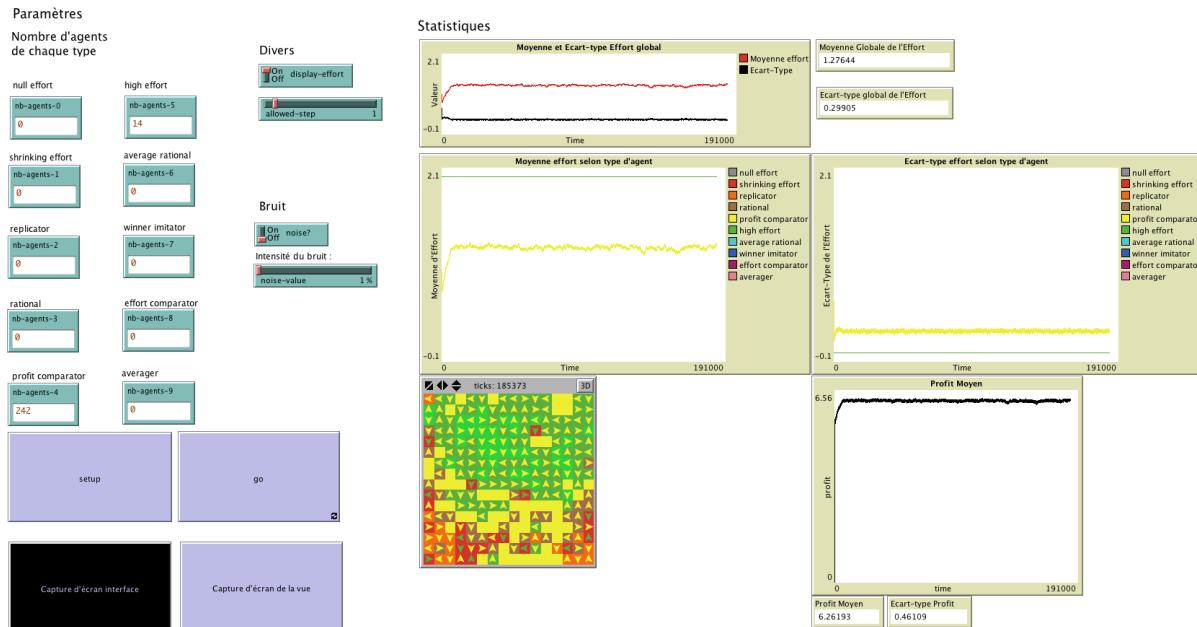


FIGURE 11 – Îlots d'effort plus élevé autour d'agents *high effort* dans une population de *profit comparator*

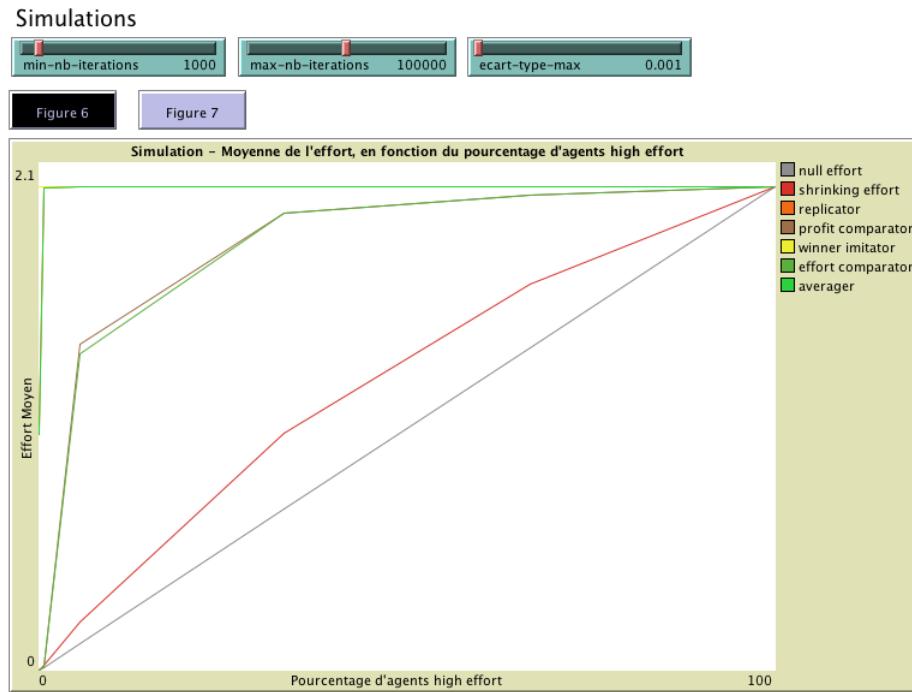


FIGURE 12 – Effet de la présence d'agents *high effort* sur une population d'agents à la rationalité limitée

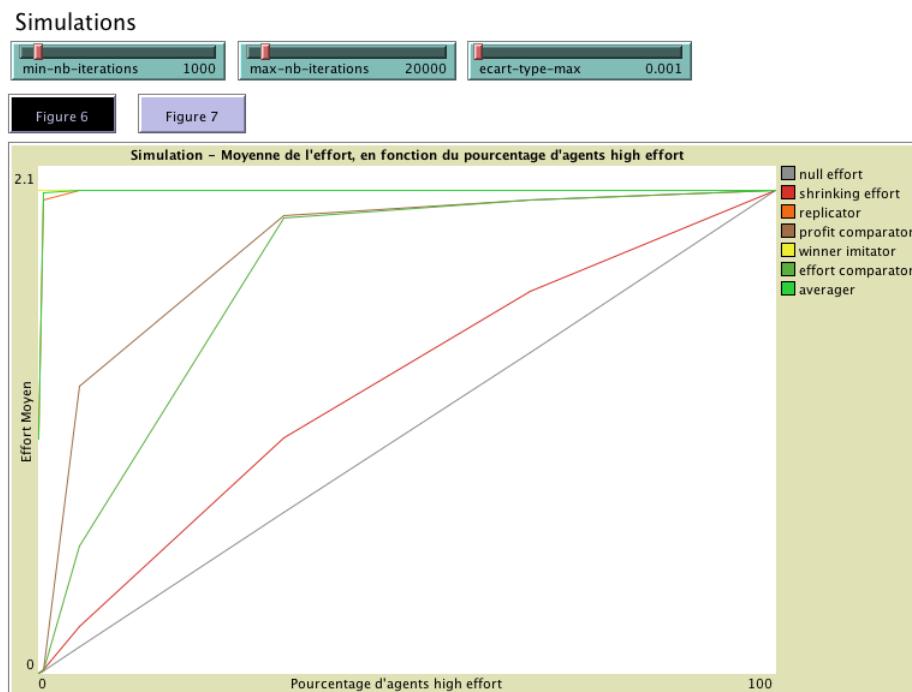


FIGURE 13 – Effet de la présence d'agents *high effort* sur une population d'agents à la rationalité limitée

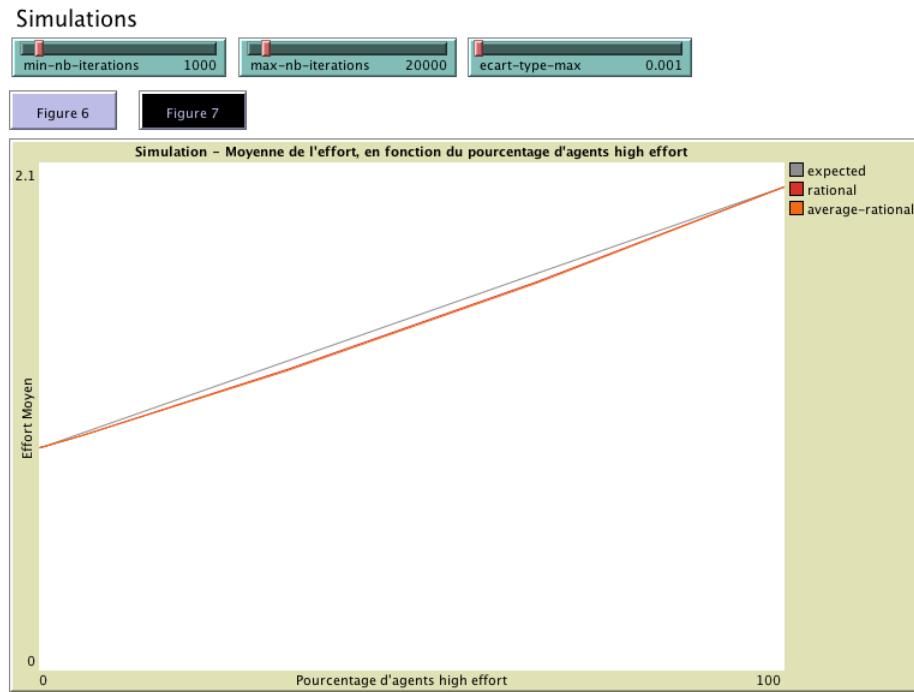


FIGURE 14 – Effet de la présence d'agents *high effort* sur une population d'agents rationnels

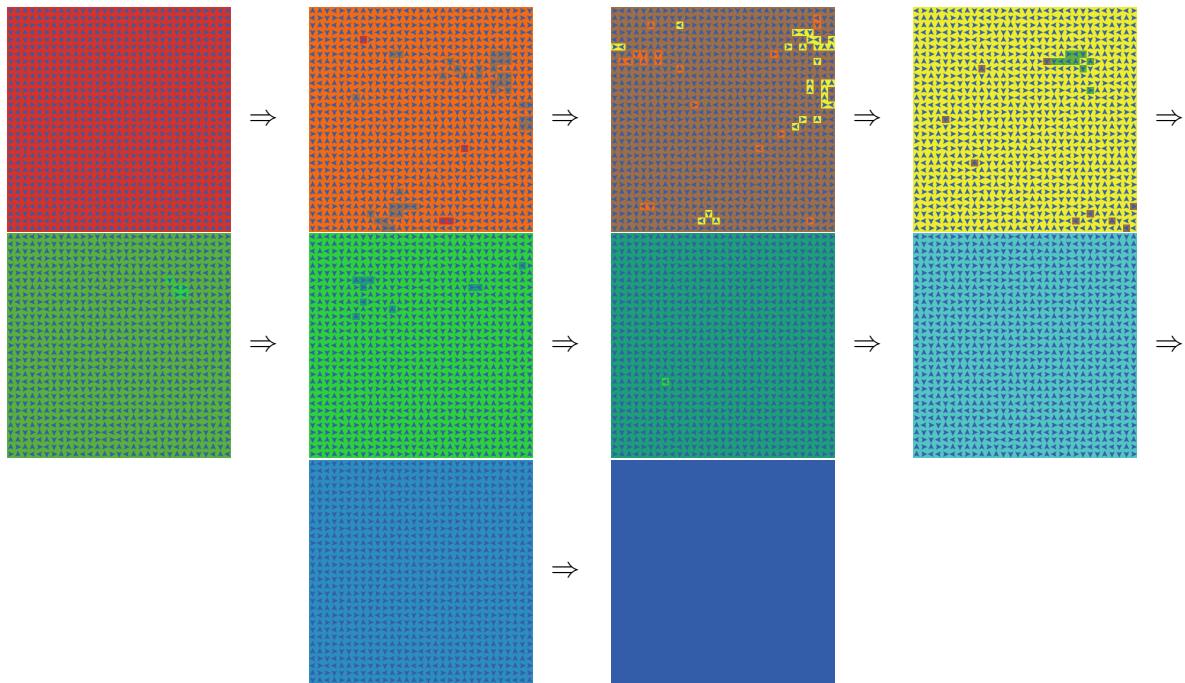


FIGURE 15 – Effet du bruit sur une population homogène d'agents *winner imitator*

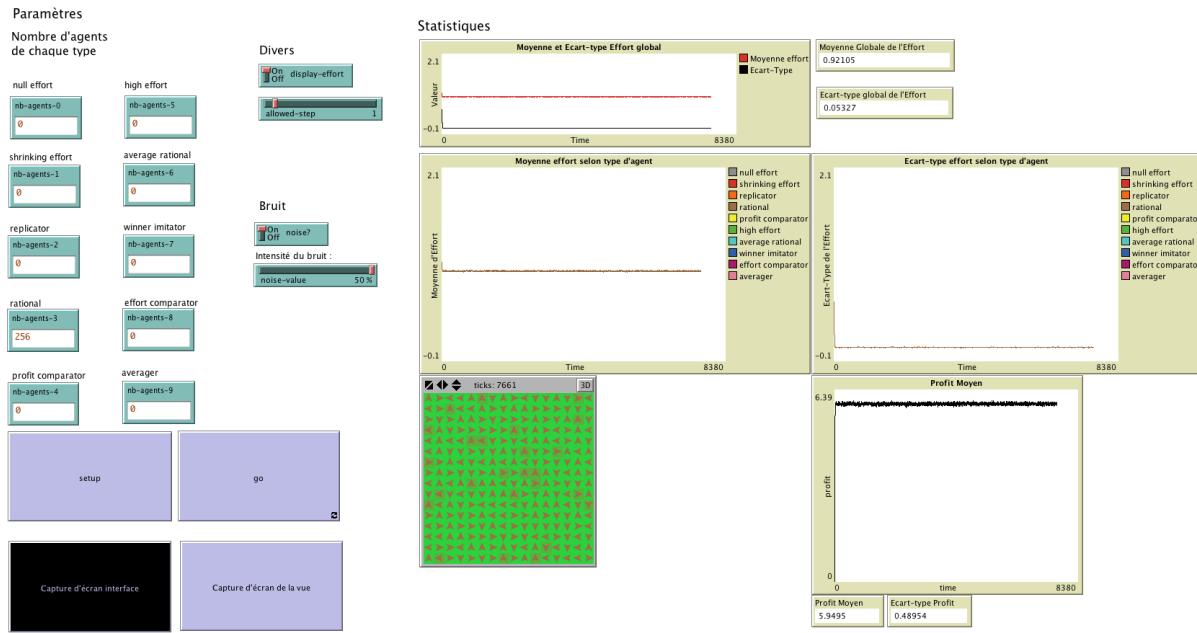


FIGURE 16 – Evolution de l'effort global d'une population de *rational*s avec 50% de bruit

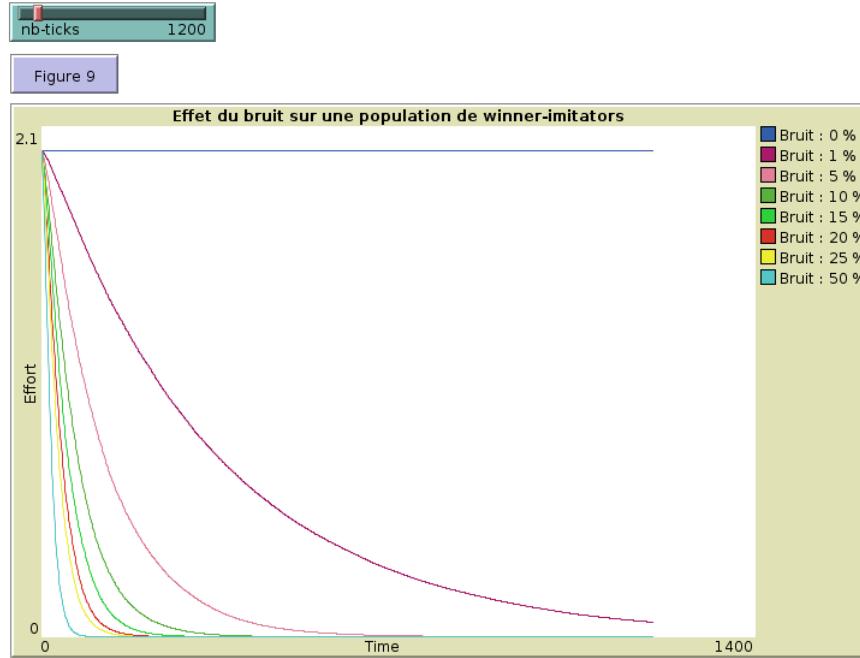


FIGURE 17 – Evolution de l'effort global d'une population de *winner imitator* suivant différentes valeurs de bruit

Améliorations

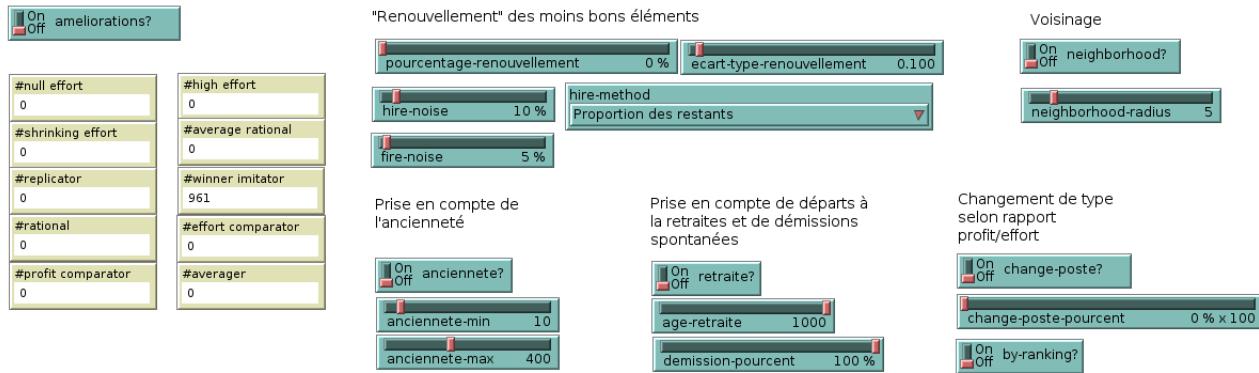


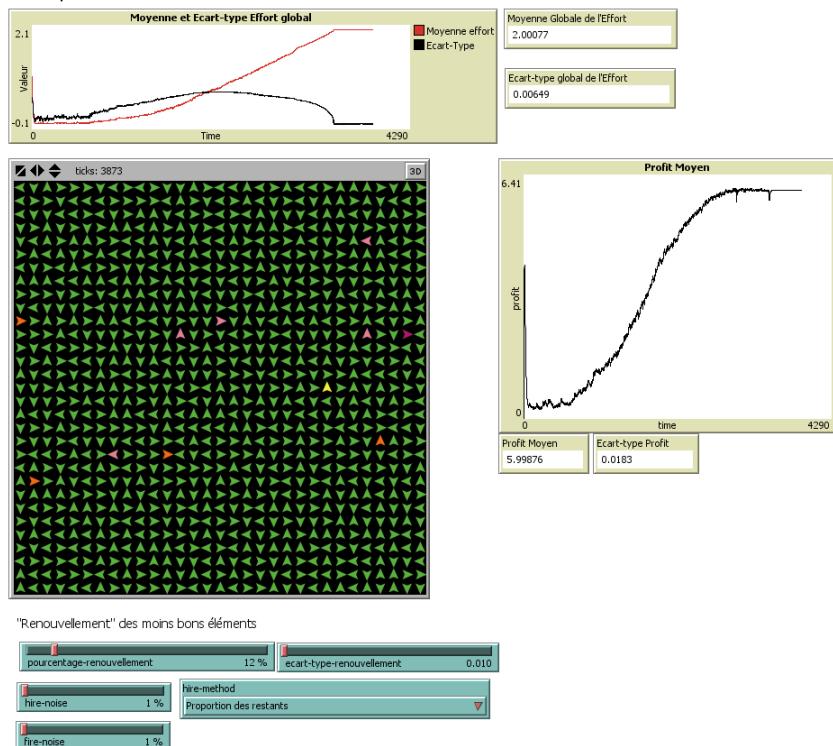
FIGURE 18 – Capture d'écran de la portion de l'interface permettant le paramétrage des améliorations

Paramètres

Nombre d'agents de chaque type

null effort	nb-agents-0	0
high effort	nb-agents-5	0
shrinking effort	nb-agents-1	961
average rational	nb-agents-6	0
replicator	nb-agents-2	0
winner imitator	nb-agents-7	0
rational	nb-agents-3	0
effort comparator	nb-agents-8	0
profit comparator	nb-agents-4	0
averager	nb-agents-9	0
#null effort	0	
#shrinking effort	0	
#replicator	4	
#rational	0	
#profit comparator	1	
#high effort	950	
#average rational	0	
#winner imitator	0	
#effort comparator	1	
#averager	5	

Statistiques

FIGURE 19 – Renouvellement d'effectif avec 1% de bruit. Population homogène de *shrinking effort*, convergence vers un ensemble d'agents de type *high effort*

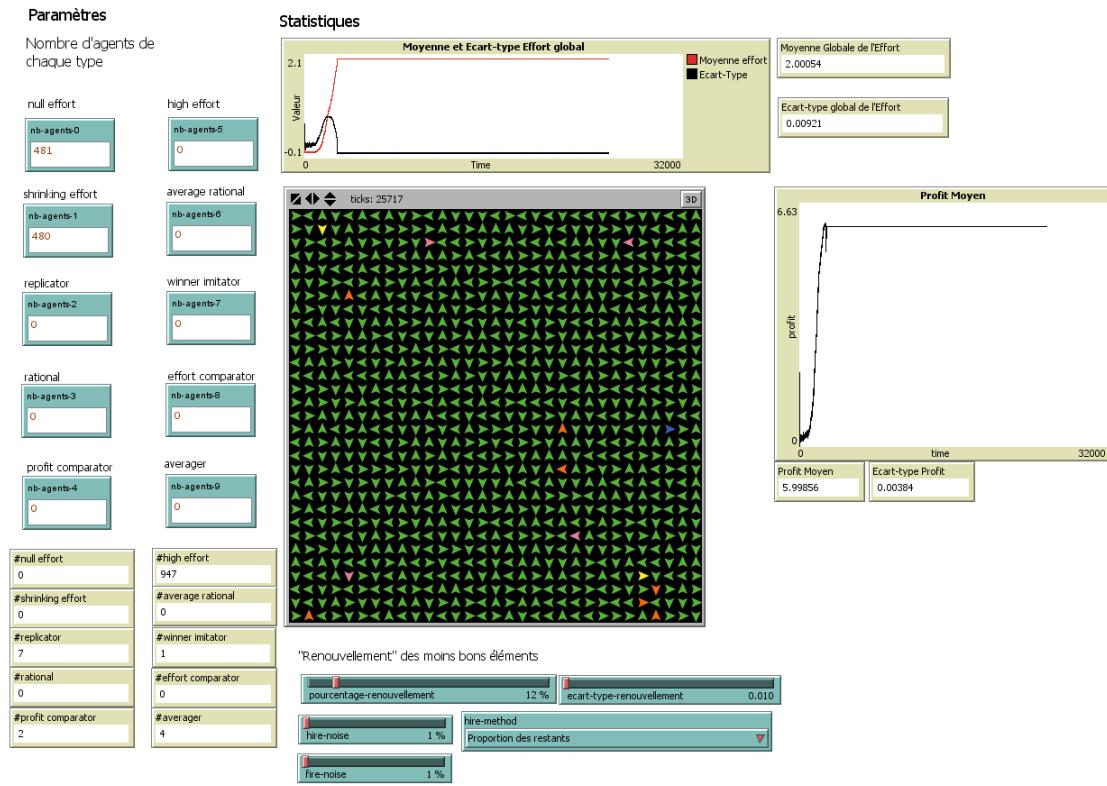


FIGURE 20 – Renouvellement d'effectif avec 1% de bruit. Population initiale de *null effort* et *shrinking effort*, convergence vers un ensemble d'agents de type *high effort*

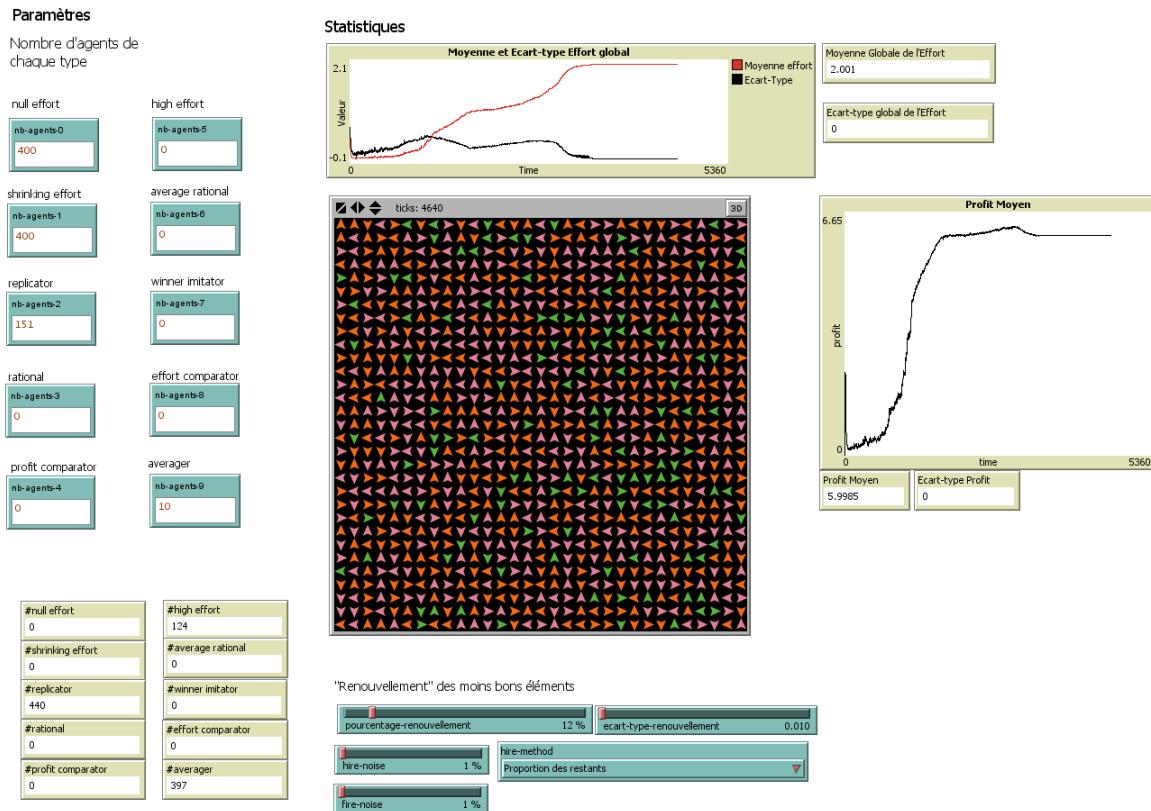
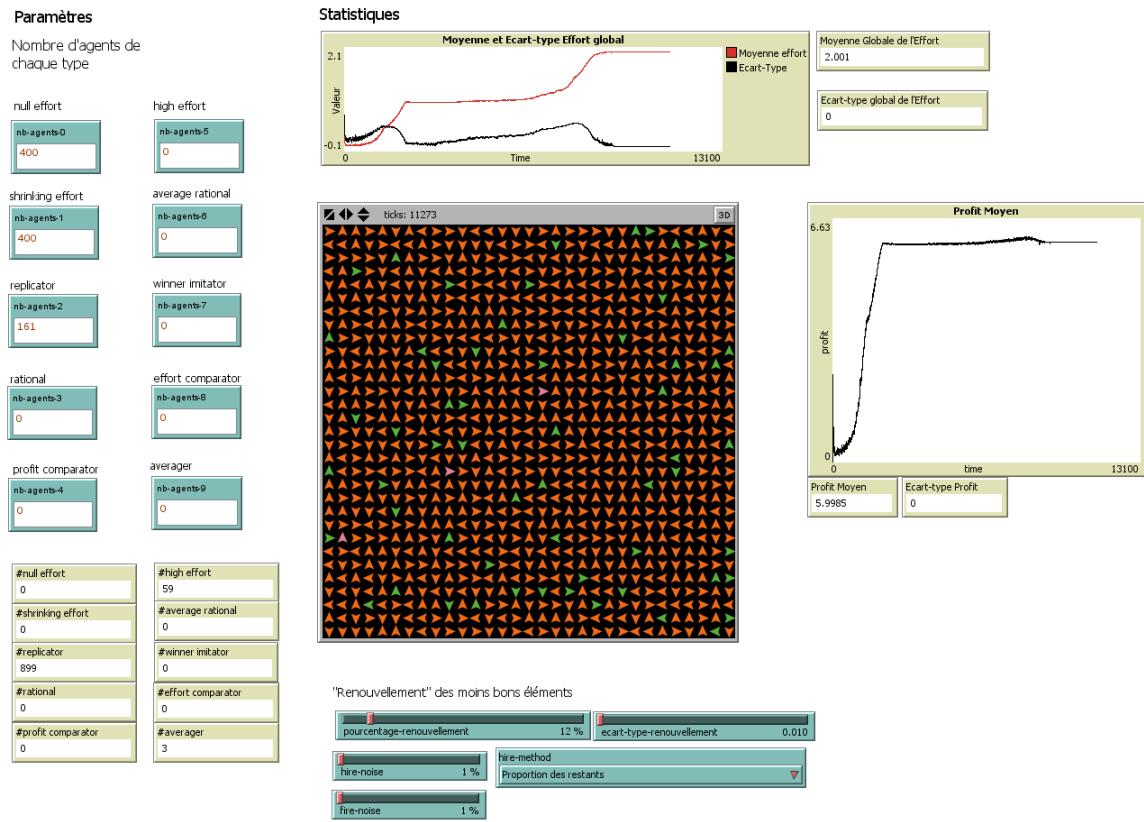


FIGURE 21 – Renouvellement d'effectif avec 1% de bruit. Population initiale hétérogène de *null effort* et *shrinking effort*, et peu de *replicator* et *averager*. Les type *replicator* et *averager* persistent.

FIGURE 22 – Renouvellement d'effectif. Convergence vers le type *replicator*.

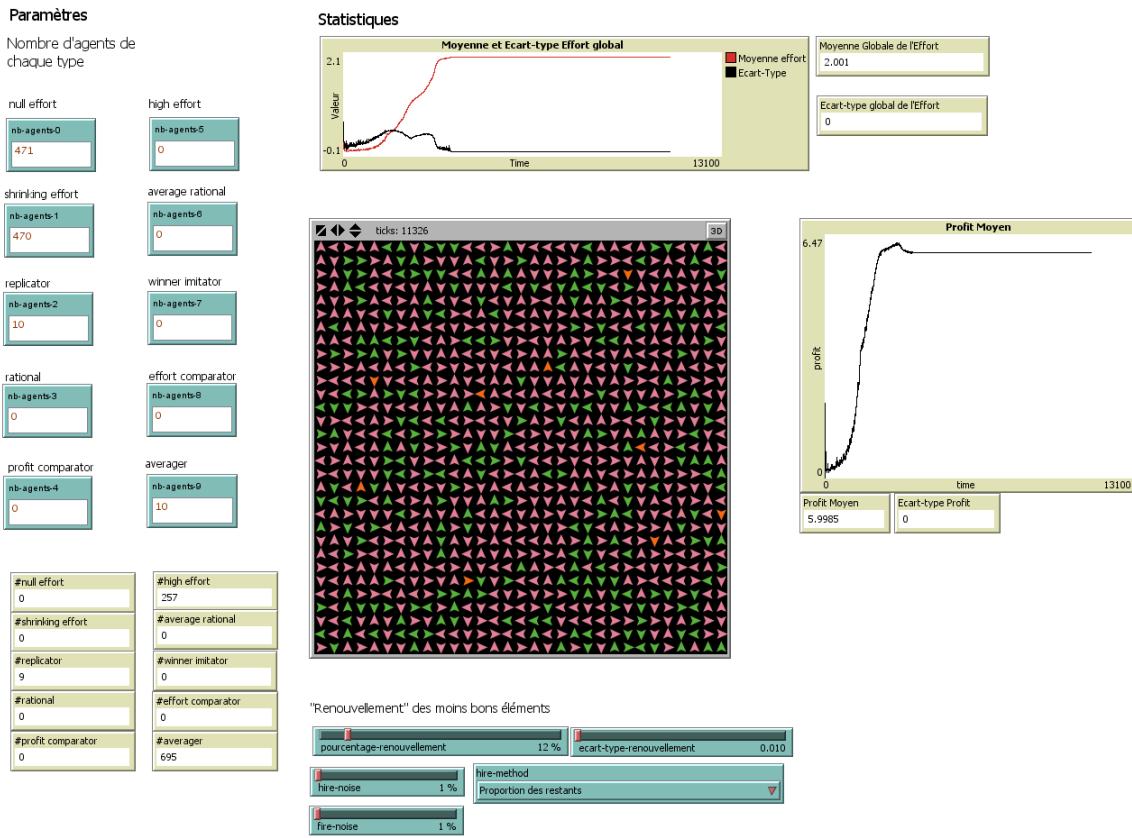


FIGURE 23 – Renouvellement d'effectif avec 1% de bruit. Population initiale de *null effort* et *shrinking effort* et peu d'*averager* (les *replicator* ne sont pas assez nombreux pour se démarquer). Les type *averager* persistent.

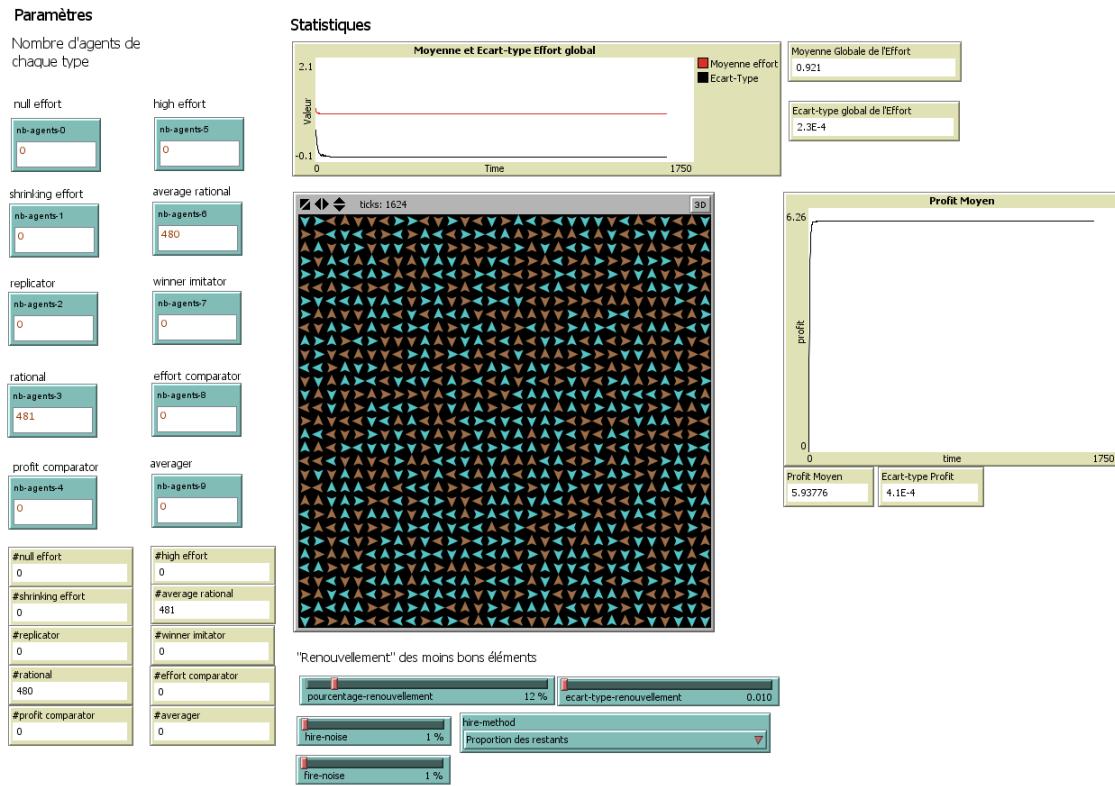


FIGURE 24 – Renouvellement d'effectif avec 1% de bruit. Population initiale de *rational* et *average rational*. Ces agents persistent, en convergeant leurs efforts vers l'équilibre de Nash.

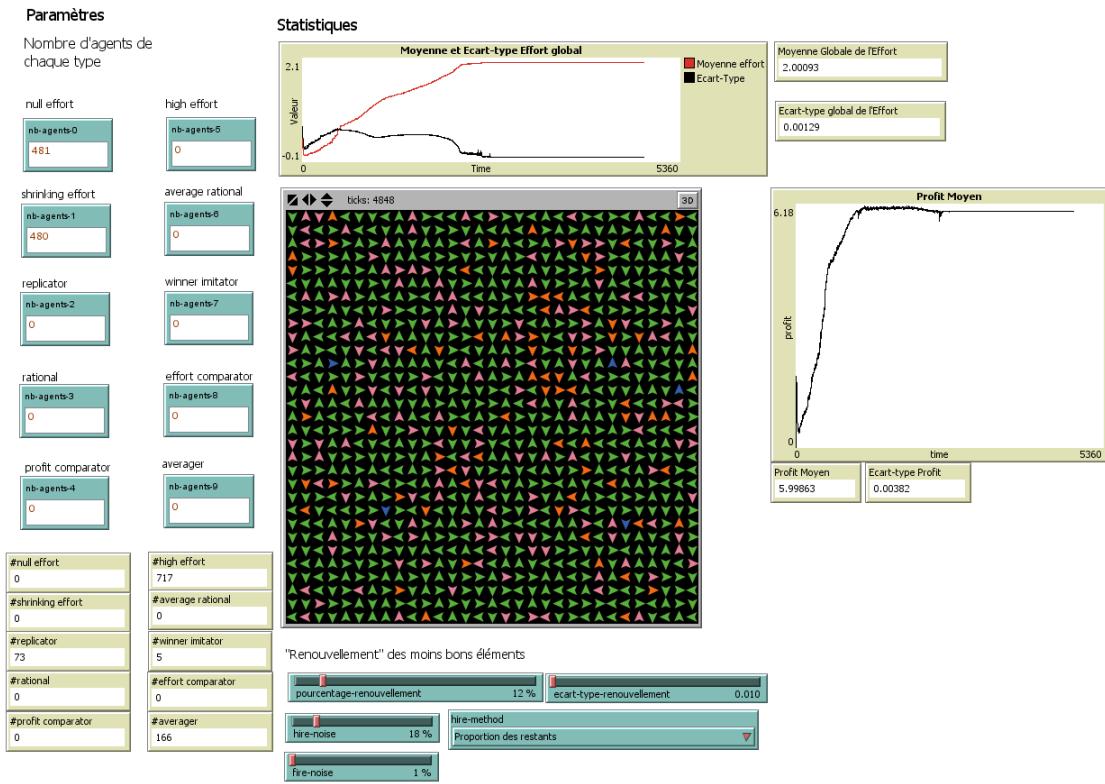


FIGURE 25 – Renouvellement d'effectif, avec 18% de bruit. Population initiale de *null effort* et *shrinking effort*. Ces agents persistent, en convergent leurs efforts vers l'équilibre de Nash.

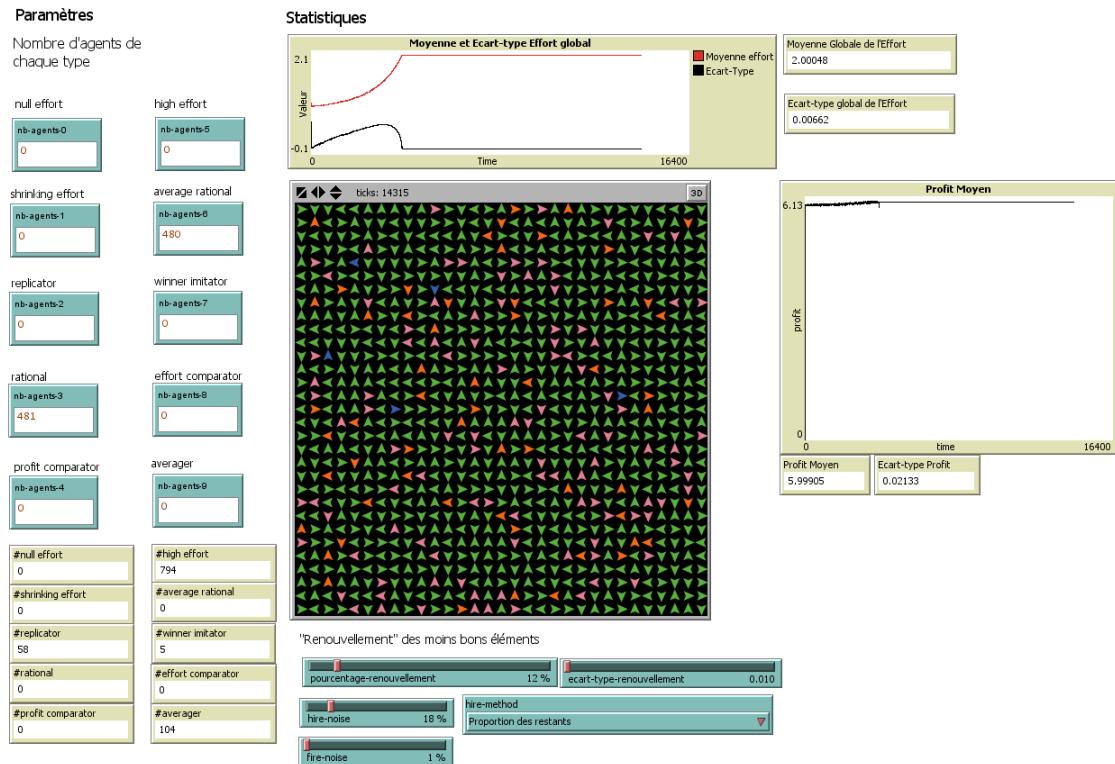
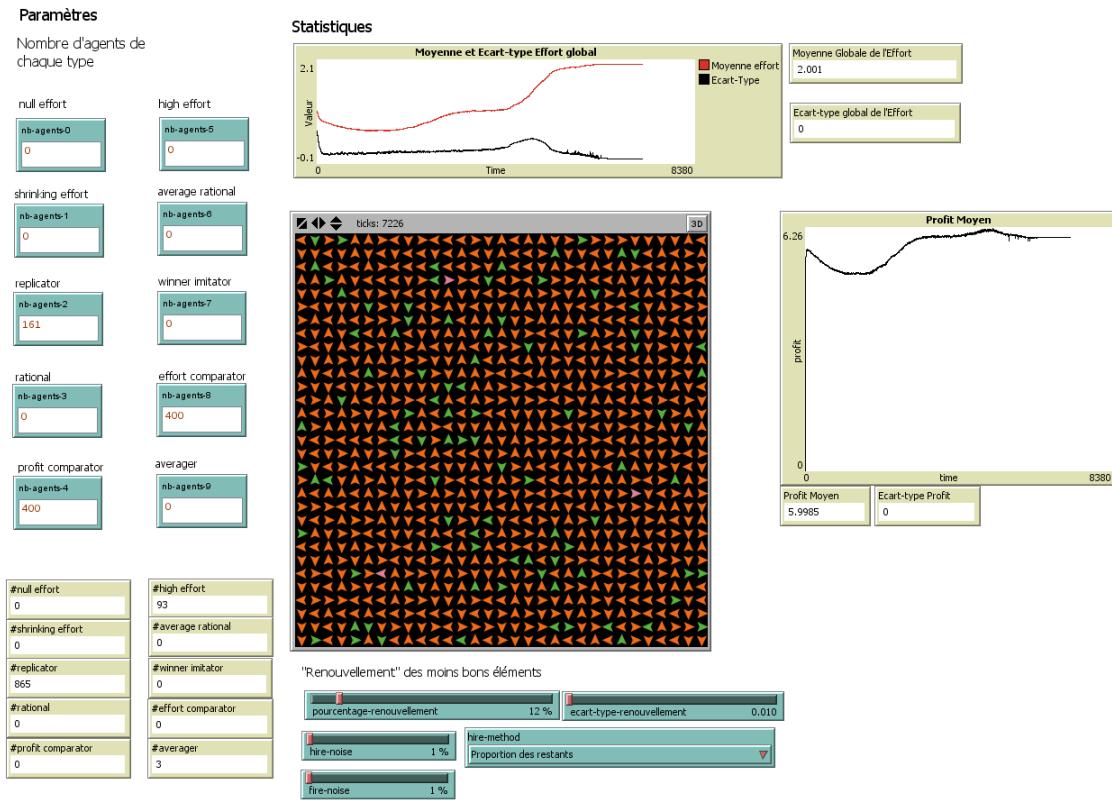


FIGURE 26 – Renouvellement d'effectif avec 18% de bruit. Instabilité de l'équilibre de Nash.

FIGURE 27 – Renouvellement d'effectif avec 1% de bruit. Convergence vers le type *replicator*.

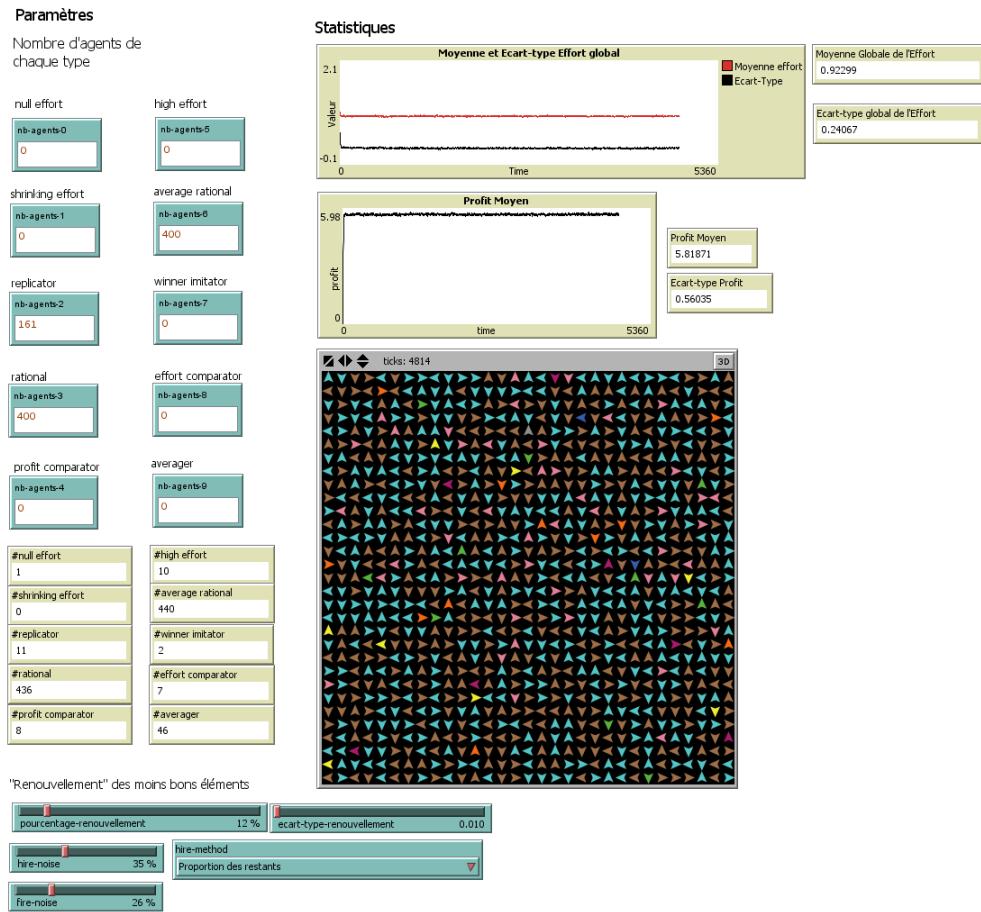
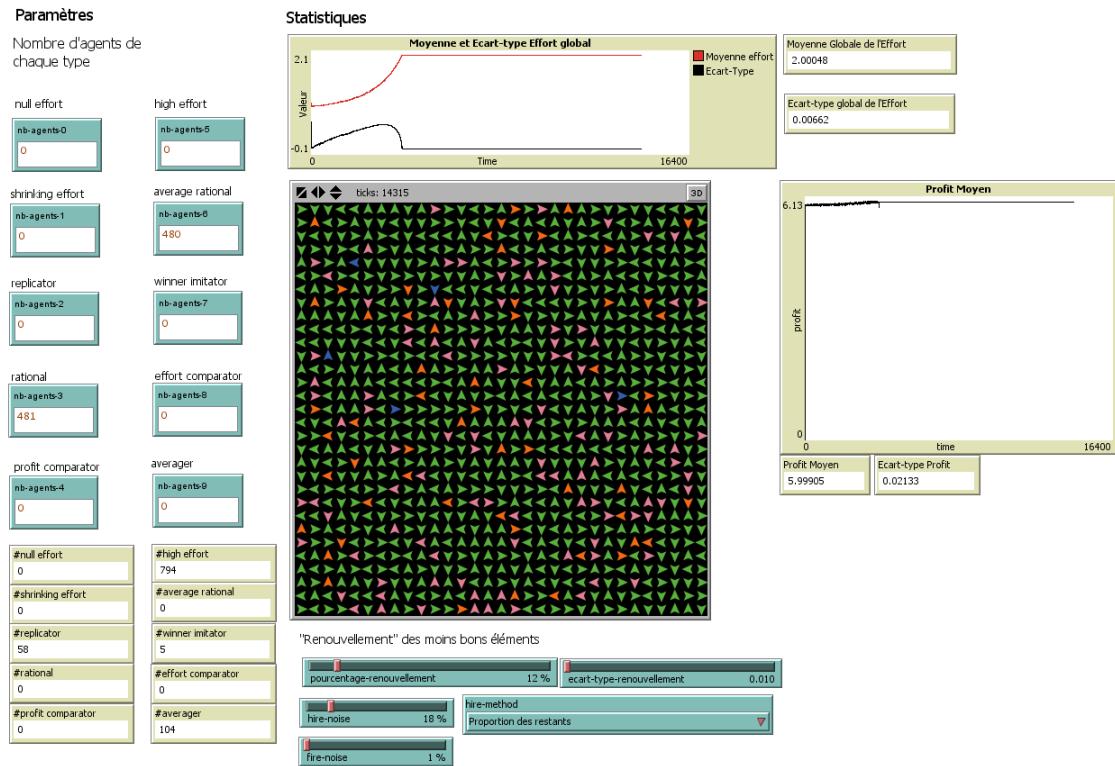


FIGURE 28 – Renouvellement d'effectif. Stabilité de l'équilibre de Nash.

FIGURE 29 – Renouvellement d'effectif. Stabilité du type *averager*.

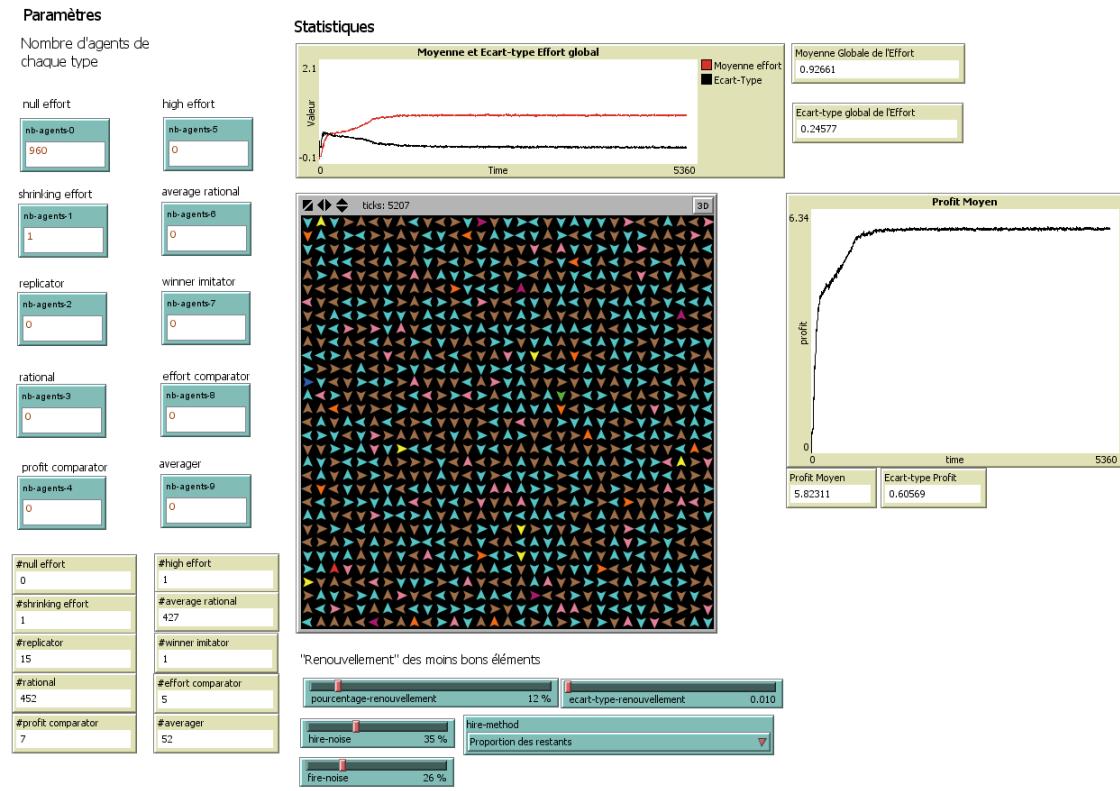


FIGURE 30 – Renouvellement d'effectif. Population initiale de *null effort* et d'un *shrinking effort*. Émergence des agents rationnels lorsqu'il y a du bruit au licenciement et à l'embauche.

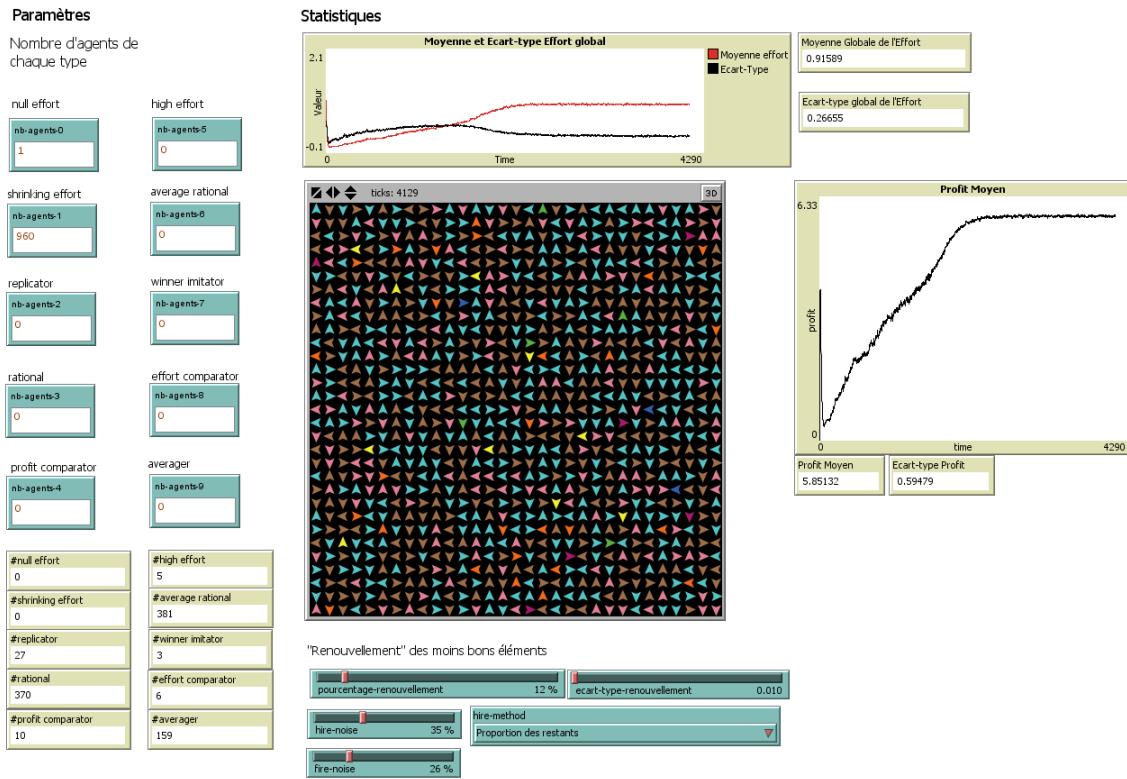


FIGURE 31 – Renouvellement d'effectif. Population initiale de *shrinking effort* et d'un *null effort*. Émergence des agents rationnels lorsqu'il y a du bruit au licenciement et à l'embauche.

A.2 Bibliographie

Références

- [1] Ariana DAL FORNO and Ugo MERLONE. A multi-agent simulation platform for modeling perfectly rational and bounded-rational agents in organizations. *Journal of Artificial Societies and Social Simulation*, 5(2), March 2002. <http://jasss.soc.surrey.ac.uk/5/2/3.html>.
- [2] D.M. Kreps. Corporate culture and economic theory. *Perspectives on Positive Political Economy*, pages 91–143, 1990.
- [3] LASSINE Alain. Résolution de l'équation du troisième degré. démonstration., Janvier 2007. http://alain.be/boece/troisieme_degre.html.