MASTER EN SCIENCES MATHÉMATIQUES

Analyse des résultats électoraux avec des modèles de dynamiques d'opinions

MORIAME, Marie

Award date:
2012

Awarding institution:
Universite de Namur

MASTER EN MATHÉMATIQUES

Analyse des résultats électoraux avec des modèles de dynamiques d'opinions

Marie Moriame

2012

Université de Namur

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX
NAMUR

Faculté des Sciences

Analyse des résultats électoraux avec des modèles de dynamiques d'opinions

Mémoire présenté pour l'obtention
du grade académique de master en
Sciences Mathématiques à finalité spécialisées

Marie MORIAME

Janvier 2012

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX
NAMUR

Faculté des Sciences

Analyse des résultats électoraux avec des modèles de dynamiques d'opinions

Mémoire présenté pour l'obtention
du grade académique de master en
Sciences Mathématiques à finalité spécialisées
Promoteur : Timoteo Carletti

Marie MORIAME

Janvier 2012

Je tiens à remercier plus particulièrement monsieur Carletti pour sa disponibilité et son soutien durant toute la durée de ce mémoire. Je tiens également à remercier mes relecteurs qui ont eu beaucoup de travail mais n'ont cependant jamais refusé. Et enfin, je tiens à remercier ma famille et mes amis qui ont du me supporter durant tout ce processus. Un tout grand merci à tous.

Résumés

L'étude des interactions entre différents individus au sein d'une même population ou groupe, est un sujet fascinant et qui a attiré récemment de plus en plus l'attention des chercheurs, surtout grâce au nombre croissant de données disponibles. Avec ce travail, nous avons étudié et proposé une modélisation du sujet en question dans le langage mathématique. Les élections politiques sont un domaine dans lequel un nombre très important des données est disponible ; pour cela nous nous sommes intéressés principalement à l'étude et la modélisation mathématique des élections, en utilisant un modèle d'échange d'opinions. En partant d'un modèle présent en littérature, nous avons mis en évidence l'existence d'une relation universelle qui lie les résultats des différentes élections de par le monde. Nous avons ensuite caractérisé le réseau social émergent du modèle avec des indicateurs standard. Finalement, nous avons généralisé le modèle en permettant aux individus de quitter ou rejoindre le groupe afin d'être encore plus réaliste.

The study of interactions between different individuals within the same population or group, is a fascinating subject and has recently attracted increasing research attention, mainly due to the number more data available. With this work, we studied and proposed a model of the subject matter in the language of mathematics. Political elections are an area where a very important data is available, why we mainly interested in the study and mathematical modeling of elections, using a model for the exchange of opinions. Starting from a model in this literature, we have demonstrated the existence of a universal relation that links the results the various elections around the world. We then characterized the emerging social network model with indicators standard. Finally, we have generalized the model by allowing individuals to leave or join the group to be more realistic.

Table des matières

Introduction

Les comportements sociaux sont des sujets très intéressants à étudier, mais il n'est pas toujours évident de trouver les données nécessaires afin de les analyser correctement. Ces différents comportements peuvent être la formation d'un consensus, la création de traits culturels communs et leur diffusion ou encore l'origine et l'évolution du langage. Il est très difficile de trouver des modèles qui permettront de modéliser ces comportements et il y a également les lois qui sont assez difficiles à extraire. Les comportements seront basés sur la communication entre les individus, et donc d'un échange d'informations mais aussi d'opinions.

Les élections peuvent être attirantes car c'est un des rares domaines où il est possible de trouver des données, notamment grâce aux différents sites internet des pays qui donnent les résultats électoraux. C'est pourquoi nous nous pencherons principalement sur ce domaine.

Nous nous baserons essentiellement sur les documents de Fortunato et Castellano ([2]) et de Banisch, Araújo et Louçã ([3]). Le premier nous permettra de dégager une relation très intéressante, qui sera applicable universellement, pour les différentes démocraties des différents pays. En effet, lors du premier chapitre, nous mettrons en évidence une distribution universelle basée sur un ré-échelonnement qui nous permettra de relier les différentes élections dans les différents pays.

Ensuite, grâce aux résultats obtenus, nous tenterons de mettre en place dans le second chapitre un modèle microscopique d'échange entre différents agents permettant d'être le plus réaliste possible et applicable aux élections durant le troisième chapitre, cette recherche sera basée sur le second article.

Enfin, nous tenterons d'être encore plus réaliste en créant un modèle ouvert où la population ne sera plus fixée, mais qui comportera un mouvement d'entrée et de sortie d'agents. Cette partie ne se basera pas sur des travaux déjà publié, mais sera tout à fait originale. En espérant en ressortir des résultats intéressants.

Chapitre 1

Distribution pour les élections proportionnelles

1.1 Introduction

Les comparaisons entre les résultats d'un modèle et les données réelles sont assez rares, mais il en existe. Notamment, nous pouvons en trouver dans les élections. Et si on regarde ces résultats électoraux d'un peu plus près, on peut pressentir une distribution dissimulée qui viendrait des votes de chaque candidat.

Afin de déterminer la distribution qui va être utilisée, nous allons comparer les comportements sociaux et dans notre cas les élections à "un comportement dynamique à grande échelle résultant de l'interaction d'un grand nombre d'atomes et de molécules" [2].

Nous pouvons dire que la distribution va être identique pour tous les pays possédant une démocratie, et indépendante de l'année de l'élection. Mais pour cela, un ré-échelonnement est parfois nécessaire. L'explication de ce ré-échelonnement fait l'objet de la section suivante.

Dans notre cas, nous nous pencherons sur les élections proportionnelles, dont on peut dire que chaque parti a son nombre de candidats qu'il présente aux électeurs et ceux-ci peuvent choisir de deux manières différentes :

1. **listes bloquées** : Chaque électeur pourra voter pour un parti, les sièges seront alors distribués proportionnellement en fonction du nombre de voix obtenues par chacun.
2. **vote préférentiel** : Chaque électeur votera pour un parti et pourra choisir un ou plusieurs candidats dans celui-ci. Le nombre variant en fonction du pays dans lequel on se trouve.

Nous nous placerons dans le deuxième cas.

1.2 Distribution

Si on prend l'exemple du Brésil, on constate que les élections dans ce pays suivent une loi $\frac{1}{v}$ où v représente le nombre de voix reçues par un candidat. Mais cette loi n'est pas universelle ; la France, l'Allemagne, l'Italie ou encore la Pologne ne la suivent pas. En effet, nous devons tenir compte du fait qu'il peut y avoir deux possibilités :

- la personne vote pour une personne d'un parti en particulier
- la personne vote juste pour le parti en lui-même.

On peut donc avoir un candidat très populaire d'un petit parti qui a le même nombre de voix qu'un candidat impopulaire d'un grand parti.

Afin d'avoir une possibilité d'étude plus étendue, nous nous pencherons principalement sur les élections "proportionnelles". Notre pays sera alors divisé en différentes circonscriptions électorales pour lesquelles sera assigné un nombre de sièges, Q_{max}.

Chaque parti recevra un nombre de sièges n proportionnels au nombre de votes qu'il aura reçu. Ces sièges seront distribués au sein du parti aux n candidats ayant reçu le plus de votes. Nous aurons alors un système basé uniquement sur le vote de l'électeur et non pas suivant la grandeur d'un parti ou d'une coalition.

Prenons un candidat i, ce candidat appartiendra au parti l_i. Q_{l_i} sera le nombre de candidats du parti et N_{l_i} représentera le nombre de votes total de l_i.

On aura alors une distribution dépendante de v, Q et N :

$$P(v, Q, N)$$

où v représente le vecteur de votes pour chaque personne, Q le vecteur de candidats par parti et N le vecteur de votes reçus par chaque parti.

Cependant, cette distribution peut être réécrite à l'aide d'une seule variable. En effet, celle-ci ne dépend pas séparément de Q et N, mais du ratio suivant :

$$v_0 = \frac{N}{Q},$$

où v_0 représente le nombre de votes moyens reçus par un candidat dans son parti. On a alors :

$$P(v, Q, N) = P_0(v, v_0)$$

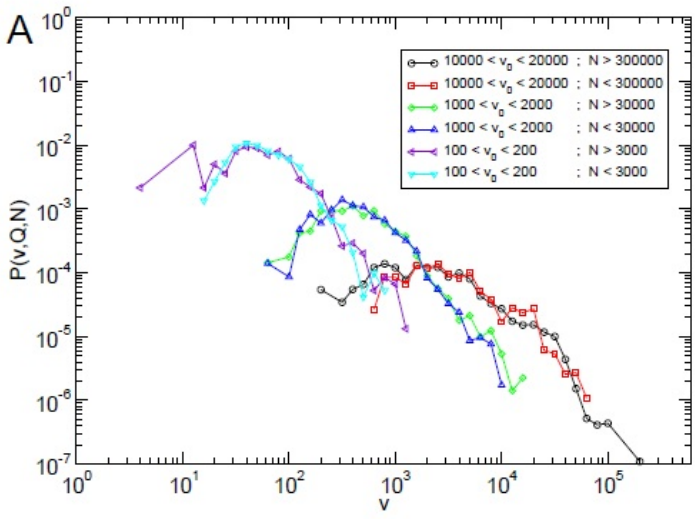


FIGURE 1.1 – Ré-échelonnement par rapport à $v_0 = \frac{N}{Q}$ pour les élections parlementaires d'Italie de 1972 [2].

On remarque que pour un v_0 fixé, la courbe ne subit pas de fortes variations contrairement à N. Ce qui nous permet de dire que la distribution dépend unique-

ment de $v_0 = \frac{N}{Q}$ et de v.

D'autre part, nous pouvons également remarquer qu'il n'y a pas de dépendance en v_0. Par contre, il y en a une en $\frac{v}{v_0}.\frac{v}{v_0}$ est considéré comme un index de performance d'un candidat par rapport à ses concurrents appartenant à la même liste que lui. On peut alors dire que si $\frac{v}{v_0} < 1$, le candidat a moins de votes que la moyenne. Par contre, s'il est supérieur, il aura plus de votes que la moyenne.

A nouveau, on effectue un ré-échelonnement et on obtient le graphe suivant.

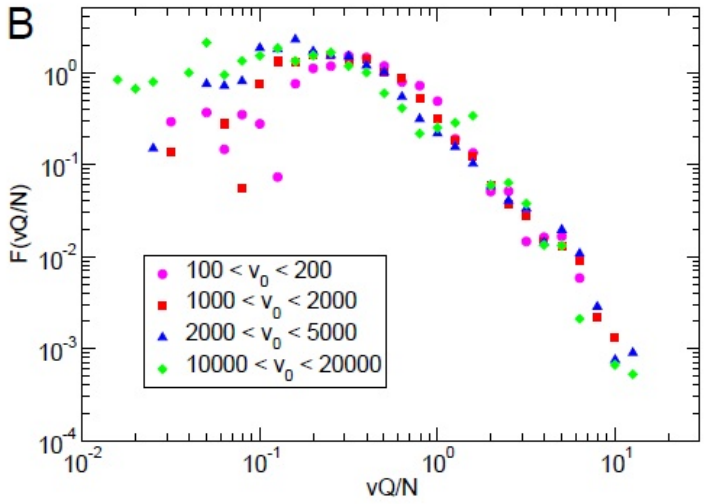


FIGURE 1.2 – Ré-échelonnement par rapport à $\frac{v}{v_0}$ pour les élections parlementaires d'Italie de 1972 [2].

On constate que les courbes sont assez similaires même en faisant varier le v_0.

Ce qui nous donne finalement :

$$P(v, Q, N) = F\left(\frac{vQ}{N}\right)$$

D'une interpolation numérique, on obtient que cette fonction suit la loi log-normale suivante :

$$F\left(\frac{vQ}{N}\right) = \frac{N}{\sqrt{2\pi}\sigma vQ} \mathrm{e}^{-\frac{\left(\log\left(\frac{vQ}{N}\right)-\mu\right)^2}{2\sigma^2}} \tag{1.1}$$

avec $\mu = 0.54$ et $\sigma^2 = -2\mu$. [2]

Remarque

Rappelons la définition d'une loi log-normale :

On a que X suit une loi log-normale si $Y = \ln(X)$ suit une loi normale de paramètres μ et σ^2.

Y suit une loi normale de paramètres μ et σ^2 si

$$f\left(y, \mu, \sigma\right) = \frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{-\frac{(y-\mu)^2}{2\sigma^2}}$$

est sa fonction de densité.

On écrira $Y \sim \mathcal{N}\left(\mu, \sigma^2\right)$, où μ est la moyenne et σ l'écart-type.

On obtient donc pour X une fonction de densité de la forme suivante :

$$f(x, \mu, \sigma) = \frac{1}{x\sigma\sqrt{2\pi}} \mathrm{e}^{-\frac{(\ln(x)-\mu)^2}{2\sigma^2}}$$

où $x > 0$, μ est la moyenne et σ l'écart-type.

On a également comme espérance attendue :

$$E(X) = \mathrm{e}^{\frac{\mu+\sigma^2}{2}}$$

[5].

Donnons une idée graphique, qui pourra nous être utile ultérieurement :

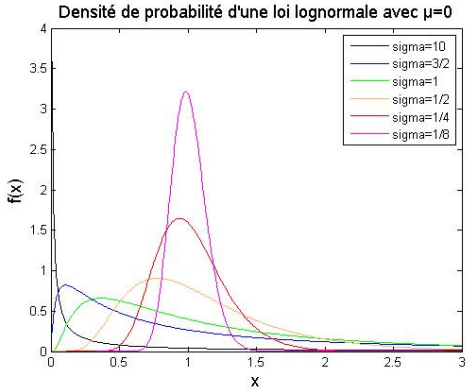


FIGURE 1.3 – Densité de probabilité d'une fonction log-normale avec $\mu = 0$

On peut donc en conclure que $\sigma^2 = -2\mu$. En effet, la valeur attendue pour $\frac{v}{v_0}$ est 1 et celle pour un log-normal est $\exp(\mu + \frac{\sigma^2}{2})$, d'où :

$$\begin{aligned} \exp(\mu + \frac{\sigma^2}{2}) &= 1 \\ &\iff \\ \mu + \frac{\sigma^2}{2} &= 0 \\ &\iff \\ \sigma^2 &= -2\mu \end{aligned}$$

1.3 Utilisations de la distribution

Utilisons maintenant les données de la Finlande pour les élections de 2003 présentes sur [6]. En effet, ce site nous permet d'avoir le nombre de votes reçus par chaque candidat pour chaque parti. En appliquant la formule 4.1 à ces données, nous obtenons le graphique suivant :

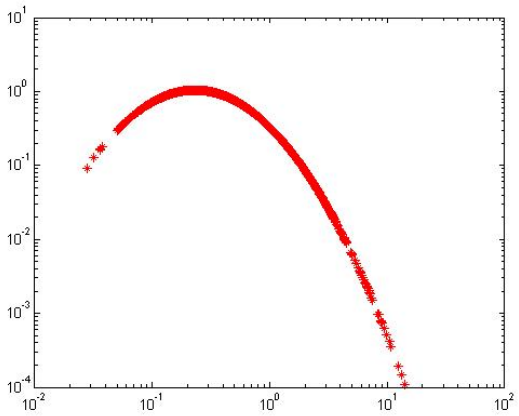


FIGURE 1.4 – $F\left(\frac{vQ}{N}\right)$ pour les élections de Finlande de 2003

Par ailleurs,[2] nous donne une représentation de la fonction de distribution pour différentes élections en Italie, en Pologne et également en Finlande. Ceci nous permet de mieux voir la distribution log-normale. On constate que toutes les élections suivent la même courbe. Ce qui confirme la bonne approximation de la distribution. Et notre bonne utilisation des données de la Finlande dans le graphique précédent.

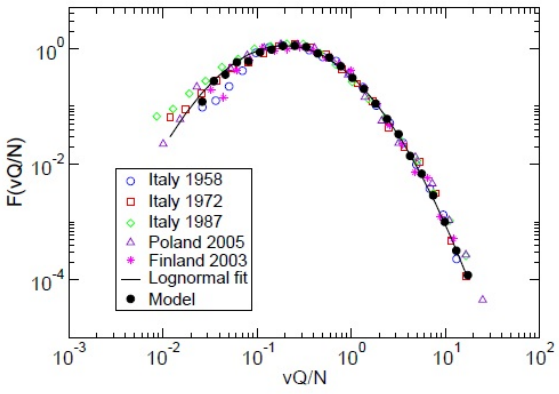
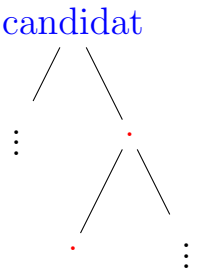


FIGURE 1.5 – Fonction $F\left(\frac{vQ}{N}\right)$ pour différentes élections, mais également pour le modèle [2].

On constate alors que toutes les élections suivent la même distribution, c'est pourquoi nous pouvons nous demander s'il n'existe pas un modèle microscopique qui permettrait d'expliquer ces résultats.

1.4 Représentation sous forme d'arbre

Le modèle d'opinion dynamique est basé sur le principe du "bouche à oreille". Chaque candidat va tenter de convaincre ses propres connaissances indécises afin que celles-ci deviennent ses porte-paroles, ce qui va créer un effet boule de neige. Nous pouvons donc considérer chaque candidat comme la base d'un arbre. La probabilité pour une personne de convaincre une des ses connaissances sera de r.



Nous aurons alors une distribution $p(k)$ qui représentera la probabilité d'avoir k connaissances, et celle-ci suivra une loi de puissance.

$$p(k) \sim k^{-\alpha} \tag{1.2}$$

où $\alpha > 1$.
Ceci est basé sur des observations de la réalité.

Remarque

Faisons un petit rappel de la loi de puissance :
On dit que X suit une loi de puissance si sa fonction de densité est de la forme

$$f(x) \sim L(x)x^{-\alpha}$$

où $\alpha > 1$ et $L(x)$ est une fonction qui va donner une petite variation et qui a comme propriété $\lim_{x \to 0} \frac{L(tx)}{L(x)} = 1$ où t est une constante.[5]
Donnons une représentation graphique de cette loi afin de mieux se la représenter :

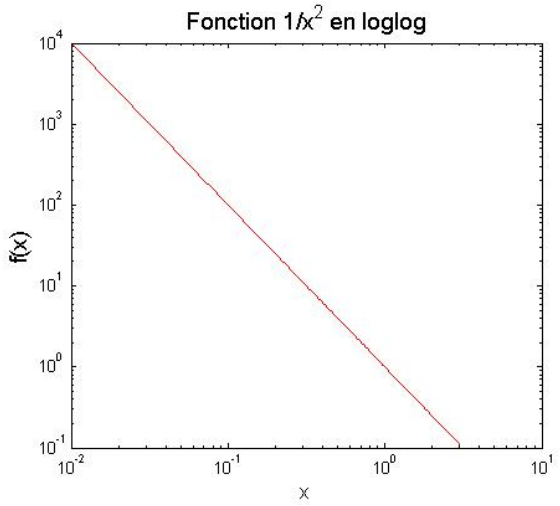


FIGURE 1.6 – Graphique de la fonction $\frac{1}{x^2}$ en loglog

Afin d'obtenir une bonne distribution, nous devrons déterminer un nombre de connaissances minimum noté k_{min}.

Nous aurons donc comme processus un candidat qui tente de convaincre ses connaissances avec une probabilité r. Ensuite, ses propres connaissances convaincues essaieront elles aussi de persuader leurs propres connaissances, jusqu'au moment où nous atteindrons un nombre de convaincus N. N représentant la taille de l'électorat du parti.

Dans [2], des simulations basées sur le principe expliqué précédemment ont été faites avec un $k_{min} = 10$, $\alpha = 2.45$ et $r = 0.25$. Nous avons pu observer graphiquement les résultats obtenus en les comparant aux données des élections de l'Italie, de la Pologne et de la Finlande (Figure 1.5).

On observe une impressionnante régularité des données, mais également une assez bonne approximation des simulations.

Nous avons donc pu constater qu'il était possible de trouver une distribution qui concordait avec les différentes élections des différents pays.

Chapitre 2

Modèle de formation d'opinion

2.1 Introduction

Beaucoup de pensées humaines peuvent être représentées sous forme de bi-polarité, c'est-à-dire oui/non, jeune/vieux, bon/mauvais, linux/windows, ou encore démocrate/républicain, $\cdots$ Cette idée va nous permettre de construire notre modèle, qui sera basé sur la proximité de pensée liant deux individus capables d'interagir. Si ces deux individus sont assez proches au point de vue de leurs convictions, alors ils seront capables d'interagir l'un avec l'autre.

C'est pourquoi, dans la suite, l'électeur sera représenté avec k-bits, c'est-à-dire un vecteur de k composantes. Celles-ci représenteront ce type de questionnement et pourront être vues sous forme binaire, c'est-à-dire 0 ou 1. Notons que par la suite, nous utiliserons -1 et $+1$ car si l'on décide par exemple, de prendre les questionnements repris ci-dessus nous leur assigneront les valeurs comme suit :

1. jeune= $+1$, vieux= -1 ;
2. bon= $+1$, mauvais= -1 ;
3. républicain= $+1$, démocrate= -1 ;
4. linux= $+1$, windows= -1 ;
5. café= $+1$, thé= -1 ;

La représentation en 5-bits d'une personne pourrait alors être :

+1	+1	−1	−1	+1

ou encore, en utilisant des bâtonnets de couleurs :

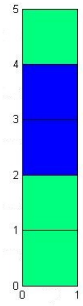


FIGURE 2.1 – Exemple de 5-bits

Dans le chapitre suivant, nous appliquerons notre algorithme dans le but de découvrir de quel candidat notre population serait la plus proche. Donc, pendant la durée de l'algorithme nous regarderons pour quel candidat chaque agent voterait en fonction de la distance de hamming que nous verrons plus tard.

Par la suite, nous tenterons d'appliquer la distribution qui a été vue auparavant sur les résultats obtenus lors de l'élection.

2.2 Modèles

Différents modèles ont été mis au point avant d'obtenir celui qui nous intéresse. Tout d'abord, il faut considérer que tous ces modèles se basent sur deux règles :

1. les individus doivent être assez proches afin de pouvoir interagir, c'est-à-dire ne pas avoir un nombre de bits différents trop important.
2. deux individus proches deviendront encore plus proches.

C'est pourquoi, on converge vers une population, où les gens partageant les mêmes opinions deviennent très proches, alors qu'ils s'éloignent de plus en plus de ceux qui ne les partagent pas. Par conséquent, si l'on part d'une société fragmentée où seulement peu de personnes partagent la même opinion, le modèle nous emmènera vers une société homogène, ou du moins avec des groupes importants qui partagent la même opinion.

Une autre caractéristique des modèles est que certains d'entre eux auront des opinions qui prendront des valeurs continues. Cependant, la plupart des modèles resteront dans le discret afin de mieux refléter les caractéristiques. Comme il a été dit précédemment, ces variables discrètes seront le plus fréquemment sous forme bipolaire.

Certains modèles étudieront la manière dont la population évolue, en étudiant par exemple, comment nous pouvons arriver à une homogénéité. Un paramètre sera mis en place afin de pouvoir relater les interactions qui ont eu lieu. D'autres modèles étudieront la possibilité qu'un agent puisse couper ses liens avec d'autres agents ...

Modèle d'opinion dynamique

Passons maintenant au modèle qui correspond à notre situation.
Nous aurons des agents sous forme de k-bits représentant leur opinion. Deux agents seront capables d'interagir l'un avec l'autre si leur nombre d'opinions différentes est plus petit ou égal à un certain seuil noté d_I.

Pour commencer, on génère N agents à qui on assigne k-bits aléatoirement. Ensuite, deux agents sont choisis aléatoirement parmi les N. Ils seront capables d'interagir s'ils sont assez similaires, c'est-à-dire, si leur nombre de bits différents est inférieur ou égal à d_I. Si ce nombre est inférieur, alors le premier agent changera un de ses bits qui était différent afin de se rapprocher de l'autre agent.
La difficulté de cet algorithme sera de trouver le d_I adéquat correspondant au k et aussi une bonne condition d'arrêt, afin d'avoir un modèle cohérent.

Prenons un exemple pour mieux comprendre le concept.

Exemple

Générons une population de 15 personnes, en leur assignant 6-bits à chacun de manière aléatoire. Nous obtenons la population suivante, où le bleu représente -1 et le vert 1.

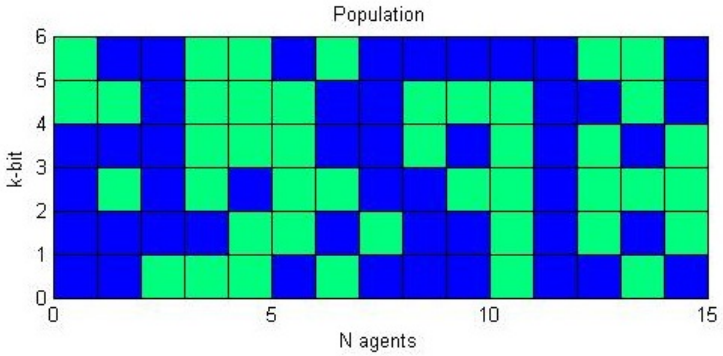


FIGURE 2.2 – Population totale

Prenons deux agents choisis aléatoirement parmi les 15 présents : $c6$ et $c8$.

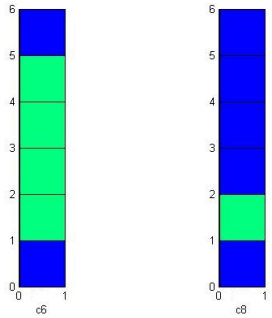


FIGURE 2.3 – Les agents choisis

On constate que ces deux agents ont trois bits en commun. On regarde alors si le

nombre de bits différents, c'est-à-dire trois, est inférieur ou égal à d_I. Introduisons une distance qui nous sera utile par la suite.

Définition 2.2.1
Soit A un alphabet et F l'ensemble des suites de longueur n à valeur dans A. La ***distance de Hamming*** *entre deux éléments a et b de F est le nombre d'éléments de l'ensemble des images de a qui diffèrent de celle de b.*

$\forall\ a,b \in F\ a = (a_i)_{i\in[0,n-1]}\ et\ b = (b_i)_{i\in[0,n-1]}\ d(a,b) = \sharp\{i : a_i \neq b_i\}$
[5]

Nous avons donc dans notre cas que si la distance de hamming entre les deux agents est inférieure ou égale à d_I ($h(c_6, c_8) \leq d_I$), alors $c6$ va changer un de ses bits différents pour se rapprocher de $c8$. Si non, les deux restent inchangés.

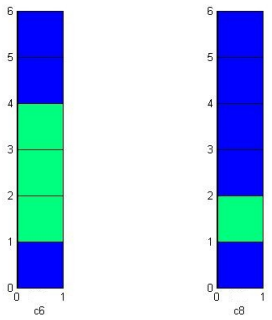


FIGURE 2.4 – Les agents choisis après changement

Une étape du modèle consiste à faire N interactions entre deux agents choisis aléatoirement c_i, c_j, où N représente le nombre d'individus (pour nous $N = 15$). Il est important de préciser que les agents ne peuvent pas interagir avec eux-mêmes, peuvent être choisis plusieurs fois ou encore pas du tout. Le processus sera arrêté lorsque il n'y aura plus d'interaction possible.

Afin de conserver une trace de l'interaction entre les agents, une matrice va alors être mise en place. Cette matrice est appelée la matrice d'interaction et est notée I :

$$I = [i_{ij}] \qquad\qquad N \times N$$

où i_{ij} représente le nombre de fois où c_i et c_j ont interagi. i_{ij} et i_{ji} seront incrémentés de 1 si $h(c_i, c_j) \leq d_I$.

Notons que l'on peut considérer la matrice I comme une matrice de poids de communication entre les différents agents.

Algorithme

Donnons à présent un résumé des étapes à suivre afin de réaliser le modèle :

1. création de manière aléatoire d'une population de N agents possédant k-bits.
2. un processus dynamique répété N fois :
   a. choix aléatoire de deux agents c_i et c_j.
   b. calcul de la distance de hamming. Si $h(c_6, c_8) \leq d_I$, alors un des bits inégal de c_i choisi aléatoirement est changé.
3. Fin du processus lorsqu'il n'y a plus d'échange possible.

En résumé, si des personnes sont assez proches pour interagir, elles le deviendront encore plus alors que si elles ne le sont pas, elles vont s'éloigner de plus en plus l'une de l'autre.

Programmation de l'algorithme

Expliquons maintenant notre algorithme de manière plus approfondie en ce qui concerne la programmation.

Reprenons étape par étape l'algorithme :

1. création de manière aléatoire d'une population de N agents possédant k-bits.

   Pour cette étape nous allons créer un tableau de k lignes et N colonnes à l'aide de la fonction "randi" qui a pour but de nous retourner un tableau $k \times N$ rempli de 0 et 1 de manière complètement aléatoire. Chaque colonne de ce tableau représentera un agent.

2. un processus dynamique répété N fois :
   a. choix aléatoire de deux agents c_i et c_j.
      A nouveau, pour cette étape nous allons utiliser la fonction "randi" pour choisir aléatoirement deux agents parmi les N.

   b. calcul de la distance de hamming. Si $h(c_6, c_8) \leq d_I$, alors un des bits inégal de c_i choisi aléatoirement est changé.

      Ensuite, nous allons créer une fonction de hamming qui sera capable de dire le nombre de bits inégaux entre deux agents mais aussi la place de ces bits. Une autre fonction un petit peu plus simple sera créée où les places ne seront pas retenues dans le cas où nous n'en avons pas besoin.

3. Fin du processus lorsqu'il n'y a plus d'échange possible.

   Ceci est la phase la plus longue de l'algorithme car elle consiste à comparer les éléments un à un, afin de trouver deux agents qui sont encore capables d'interagir. Dans le but de simplifier cette phase mais aussi peut-être d'accélérer notre algorithme, nous utilisons la fonction "unique" dans matlab. Cette fonction ressort un tableau comprenant chacun des agents différents. L'avantage de ce tableau est que chaque agent n'est plus représenté qu'une seule fois, ce qui nous permet donc d'obtenir des boucles moins longues. C'est dans ce cas-ci que nous prenons la fonction de hamming simplifiée.

Phases de l'algorithme

Lors de l'itération, le modèle va passer par trois phases :

1. **"burn-in-phase"** : premier moment où il n'y a pas encore de déviation significative.
2. **"transient-phase"** : moment entre l'aléatoire et l'ordre.
3. **"final dynamic"** : moment où tous les agents vont converger vers un état stable, et ce, assez rapidement.

Nous obtiendrons évidemment des résultats très différents suivant les valeurs données à d_I. En effet, si on prend un d_I assez élevé, nous convergerons très rapidement vers un consensus. Puisque nous regardons le nombre de différences qu'il existe entre deux individus, ils interagiront plus facilement l'un avec l'autre si on leur permet davantage de différences.
Au contraire, si le seuil est très bas, deux agents auront beaucoup de difficultés à interagir l'un à avec l'autre, excepté s'ils sont très proches.

2.3 Les différents comportements possibles

Nous aimerions obtenir un modèle le plus proche possible de la réalité, c'est pourquoi il convient de trouver un d_I en adéquation avec le k. Il serait également intéressant de considérer un modèle où le nombre de paramètres est réduit afin de ne pas avoir un trop grand nombre d'itérations à exécuter.
Nous testerons les comportements du modèle suivant les valeurs de k et d_I. Pour cela nous prendrons $k = 1 \cdots 32$ et $d_I = 1 \cdots k$.

Essayons tout d'abord de classifier les différents groupes de comportement que nous pourrions obtenir.

Introduisons N_G le nombre de groupes d'individus qui partagent exactement la même opinion et

$$G_a = \{c_i : h(c_i, a) = 0\}$$

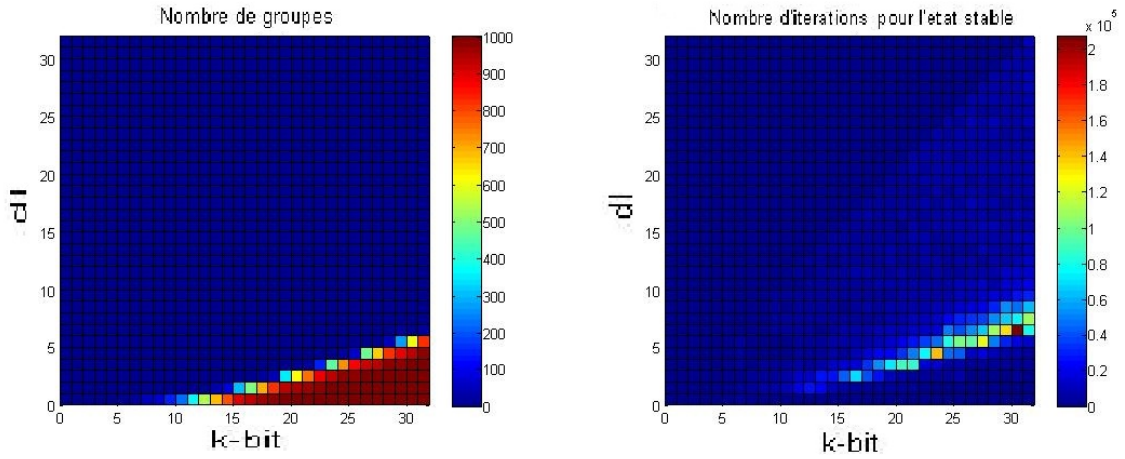
où a représente l'opinion partagée par tous les c_i du groupe.

On peut donc reconsidérer N_G comme le nombre de groupes qui contiennent au moins un membre. Ce qui nous donnera alors

$$\max(N_G) = \min(N, 2^k).$$

En effet, étant donné que l'on a k-bits on a forcément 2^k possibilités, mais on ne peut pas avoir plus de N individus dans N_G.
On peut alors différencier deux cas pour N_G. Soit tous les individus ont la même opinion et ils sont donc tous dans le même groupe. Dans ce cas, $N_G = 1$ (consensus). Ou alors tous les individus ont une opinion différente, ils forment chacun leur groupe et $N_G = N$ (fragmenté).

Dans les graphes suivants, nous regardons les différents comportements du modèle pour $k = 1 \cdots 32$ et $d_I = 1 \cdots 32$, avec une population de 1000 agents.
Pour cela, nous avons utilisé matlab, en faisant tourner notre algorithme suivant les différents k et d_I. Ensuite, nous avons regardé le nombre de groupes ayant les mêmes opinions et donc les mêmes agents, c'est-à-dire N_G. Mais, nous avons également regardé le nombre d'itération nécessaire à la réalisation de l'algorithme dans les différents cas. Ce qui nous permet d'obtenir les figures suivantes :



(a) Nombre de groupes à l'état stable (b) Nombre d'itérations pour arriver à l'état stable

FIGURE 2.5 – La figure de gauche représente le nombre de groupes qui sont formés à la fin de l'algorithme et la figure de droite désigne le nombre d'itérations nécessaires à la réalisation de cette algorithme.

Que pouvons-nous déduire de ces deux premiers graphiques ?

Tout d'abord, nous pouvons voir une première partie du graphique qui représente une population identique, c'est-à-dire qui est arrivée à un consensus et qui ne forme qu'un seul et unique groupe. Cette partie étant représentée en bleu.

Ensuite, nous pouvons voir une population très fragmentée où le nombre de groupes va être de 1000, c'est-à-dire que chaque agent a une opinion différente. Cette partie étant représentée en rouge foncé.

Et enfin, la partie la plus intéressante pour nous, la partie qui sépare les deux autres situations, c'est-à-dire le moment où on peut voir des groupes de plusieurs agents se former afin d'être dans un état entre le consensus et la fragmentation. Ceci représente mieux la réalité. On peut voir que ce changement commence à s'opérer lorsque l'on passe $k = 10$, c'est-à-dire que le nombre maximal de groupes, qui est donné par $\max(N_G) = \min(N, 2^k)$, correspond à $min(1000, 1024)$.

Passons maintenant à l'analyse du graphique qui représente le nombre d'itérations nécessaires pour arriver à l'état stable.

On peut voir que nos deux phases extrêmes, c'est-à-dire celles représentées en bleu et en rouge dans le graphique précédent sont, dans ce cas-ci, les situations qui possèdent le moins d'itérations et sont donc représentées en bleu.

En ce qui concerne la situation intermédiaire, on constate un nombre d'itérations très élevé de l'ordre de 10^5. En sachant que dans chaque itération le processus d'échanges entre deux agents est répété N fois. Et dans notre cas, la population n'est que de 1000 individus, ce qui pourrait poser des difficultés. Par exemple, dans des cas où la population serait celle d'un pays qui se compterait en millions.

2.4 Analyse du seuil

Nous allons à présent représenter la taille des différents groupes qui vont être obtenus à la fin de la simulation. Cela nous permettra de mieux analyser l'algorithme, mais également de décider quel d_I sera nécessaire dans les différentes applications de notre algorithme. Pour cela, nous allons prendre un nombre d'agents égal à 1000 et un $k = 20$. Nous exécuterons alors l'algorithme avec un d_I variant de 1 à 5 et nous ferons trente fois la simulation afin de conforter nos résultats.

Ce qui nous donne le graphique suivant :

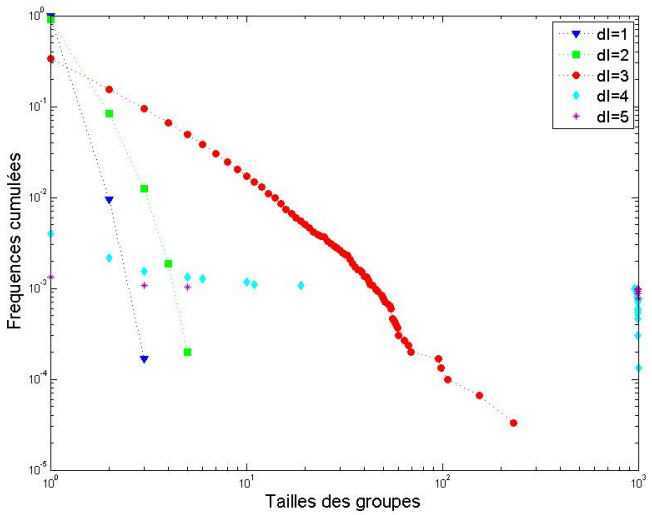


FIGURE 2.6 – Fréquences cumulées de la distribution des tailles des groupes pour $k = 20$ et $d_I = 1..5$

Analysons notre graphique suivant les différents seuils utilisés.

$d_I = 1$

Dans ce premier cas, nous pouvons voir que la plupart des groupes sont composés d'une seule personne. Nous avons donc une population très fragmentée. Ce

qui paraît assez logique, de par le fait du seuil qui est très petit.

En effet, avec un seuil de 1, cela signifie que pour que deux agents puissent interagir ils doivent avoir au maximum un bit différent sur les 20. Nos agents auront donc beaucoup de difficultés à interagir l'un avec l'autre. Cela nous donne une situation avec des groupes composés d'une seule personne et pouvant atteindre au maximum trois personnes, mais avec une fréquence très faible.

$d_I = 2$

Passons maintenant à notre deuxième seuil, il ne semble pas être vraiment différent du premier. Nous avons une petite augmentation de la taille des groupes, mais nous restons toujours dans une population très fragmentée. Cet effet est à nouveau causé par ce seuil trop faible qui ne permet pas aux différents agents d'interagir de manière active.

$d_I = 3$

Nous pouvons voir pour $d_I = 3$ une évolution dans notre modèle. En effet, celui-ci reprend des tailles de groupes assez variées. Nous pouvons voir des groupes très petits ne reprenant qu'une seule personne, mais nous avons également des groupes de presque 300 agents.

Ce seuil est également intéressant car il nous permet d'observer qu'une loi de puissance se dessine. Ceci est très intéressant puisque cela confirme ce qui avait été dit dans le premier chapitre.

$d_I = 4$

Nous voyons à présent un grand changement puisque nous avons maintenant des groupes qui ont atteint les 1000 agents et nous mènent donc à un total consensus ou du moins presque total.

Mais, nous pouvons également voir qu'il y a des groupes qui se forment d'une vingtaine de personnes. En effet, dans certains cas, si la population de départ possède des agents très différents l'un de l'autre on peut être confronté à un problème. Même avec un seuil de 4, celui-ci est encore trop élevé et donc notre population n'arrive pas à un consensus total. On se retrouve donc avec un groupe d'une vingtaine de personnes qui se détache du reste de la population. Bien sûr, ce cas n'est pas très fréquent.

$d_I = 5$

Et enfin, notre dernier seuil nous montre des modèles qui se sont pour la plupart terminés dans un consensus global avec des groupes de 1000 personnes. Ce qui est très intéressant pour la suite de notre travail. Car lorsque nous voudrons travailler dans un système qui se dirige vers un consensus, nous pourrons reprendre des paramètres du même type avec un seuil $d_I = 5$.

Ces deux derniers résultats sont assez logiques puisque le seuil étant plus élevé il permet aux différents agents entrant en interaction d'avoir des différences plus élevées. Il sera donc plus facile pour deux agents d'entrer en communication.

2.5 Réseaux de communication

Notre modèle peut aussi être considéré comme un réseau, où les différents agents représenteraient les nœuds et les arêtes, les liens qu'il y a entre deux agents. C'est pourquoi, on introduit la matrice d'adjacence A suivante :

$$\begin{aligned} A &= [a_{ij}] \\ &= 0 \quad \text{si} \quad i_{ij} = 0 \\ &\quad\ 1 \quad \text{si} \quad i_{ij} > 0 \end{aligned}$$

où i_{ij} représente les éléments de la matrice d'interaction I.

La matrice montrera s'il y a eu interaction ou non entre deux agents, mais elle ne montrera pas le nombre de fois où il y en a eu échange (la fréquence) contrairement à la matrice I. Nous pouvons donc dire que cette matrice A nous renseignera sur le degré de chaque agent, c'est-à-dire le nombre d'agents avec qui celui-ci a eu des contacts. Nous allons donc l'utiliser dans ce cas-ci afin de connaître le nombre de personnes avec qui un agent a interagi en sommant les lignes de cette même matrice A.

Afin de réaliser notre graphique de degré des nœuds, nous prendrons 1000 agents et $k = 20$. A nouveau, nous étudierons l'algorithme pour $d_I = 1..5$ et nous exécuterons le programme 10 fois.

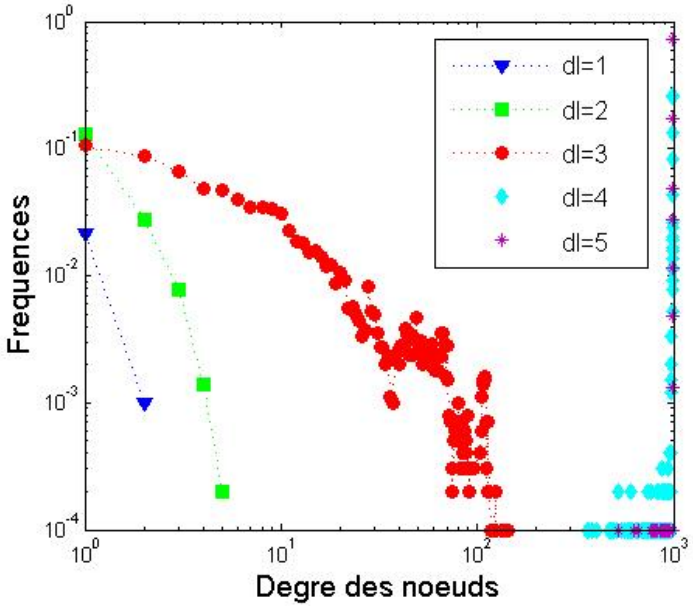


FIGURE 2.7 – Fréquences de la distribution du degré des noeuds pour $k = 20$ et $d_I = 1, \cdots 5$

Analysons notre graphique suivant les différents seuils.

$d_I = 1$

Cette distribution nous montre que pour un $d_I = 1$ le nombre de personnes avec qui les agents ont interagi est de 1 ou 2 avec une fréquence de 0.02, ce qui nous apprend que la plupart des agents n'ont interagi avec personne. Nous avons donc beaucoup d'agents isolés. A nouveau, cela semble très logique. Le seuil étant trop faible, les différents agents ont beaucoup de difficultés à interagir l'un avec l'autre.

$d_I = 2$

Pour un $d_I = 2$, nous avons à nouveau beaucoup d'agents isolés, mais les agents qui ont interagi ont jusqu'à six interlocuteurs. Le seuil étant moins stricte, il semble assez normal qu'il y ait un petit peu plus d'interactions.

Bien sûr cela reste assez faible, puisque nous nous retrouvons dans une population de 1000 personnes et nous sommes capables d'interagir uniquement avec six personnes au maximum. Cela reste très faible.

$d_I = 3$

Nous nous trouvons à présent dans notre état intermédiaire entre les agents qui ont énormément de relations et ceux qui n'en ont quasi pas. Nous pouvons voir pour $d_I = 3$ une loi de puissance se dessiner si on tient compte des connections inférieures à 100. Bien sûr, cela conforte ce qui a été dit dans le chapitre 1. Cela signifie que lorsque nous effectuerons le ré-échelonnement, nous pourrons trouver des comportements similaires entre les différentes élections.

Nous pouvons également nous dire que le comportement avec $d_I = 3$ peut être associé à la phase de transition, c'est-à-dire avec des petits groupes mais aussi de gros groupes. Nous sommes entre deux états, une population fragmentée ($d_I = 1, 2$) et une population où l'on retrouve un consensus ($d_I = 4, 5$).

$d_I = 4$

Par contre, si l'on considère un $d_I = 4$, on a des agents qui ont au minimum 300 autres agents avec qui, ils ont interagi. Nous avons donc une très grande communication, ce qui paraît tout à fait logique puisque le seuil est assez élevé. Les agents ont alors plus de facilités à échanger leurs idées et donc à interagir. En effet, notre condition $h(c_i, c_j) \leq d_I$ a plus de chance d'être satisfaite.

Maintenant, si on compare les résultats obtenus avec le graphique 3.1 réalisé précédemment, nous pouvons constater que pour un $d_I = 4$, il y avait des agents qui formaient à eux seuls un groupe. Ces agents ont au minimum interagi avec 300 autres, ce ne sont donc pas des agents isolés mais plutôt des outsiders, ils n'ont pas subi la pression des gros groupes. Donc nous pouvons avoir une population qui se dirige vers un consensus mais qui dans certains cas garde certaines personnes avec une opinion différente.

$d_I = 5$

Ce dernier seuil donne une population où les agents n'ont pas moins de 400 contacts. Cela nous donne une population qui a beaucoup de contacts et donc qui échange ses idées et se rapprochent l'un de l'autre pour obtenir une population qui partage exactement les mêmes caractéristiques.

2.6 Coefficient de clustering et diamètre

Passons maintenant à une nouvelle analyse de notre méthode en étudiant le coefficient de clustering et le diamètre d'échange dans notre population.

Pour cela, nous allons utiliser notre matrice d'adjacence A qui nous permet de savoir avec combien d'agents un individu a interagi mais aussi avec quels agents.

Tout d'abord, nous allons donner une brève explication sur le coefficient de clustering, mais aussi sur le diamètre d'une population.

Coefficient de clustering

Nous savons que notre modèle peut être considéré comme un réseau où nos agents représentent les nœuds et où l'interaction entre ceux-ci est représentée par les arêtes. On peut donc employer les méthodes utilisées pour tous réseaux. Le coefficient de clustering ou coefficient d'agglomération en fait partie.

Revenons maintenant à notre coefficient de clustering, ce coefficient représente la vraisemblance que deux associés à un nœud soient associés entre eux. Un coefficient élevé indique une tendance à faire un groupe élevé. Ce qui signifie dans notre cas que plus le coefficient sera élevé, plus il y aura de connexions entre les agents. Donc, dans la plupart des cas avec un coefficient élevé, nous pourrons correspondre au cas "les amis de mes amis sont mes amis".
Donnons une définition plus mathématique du coefficient de clustering.

Définition 2.6.1
Le coefficient de clustering est donné par :

$$C_v = \frac{2E_v}{d_v\,(d_v - 1)} \tag{2.1}$$

où

- v = *le sommet dont on veut connaître le coefficient.*
- E_v = *le nombre de liens entre les voisins de v.*
- d_v = *le nombre de voisins de v.*

[4]

Nous avons un coefficient de 2 au numérateur car nous sommes dans un cas où notre graphe est non-orienté. Dans le cas contraire, nous n'aurions pas ce coefficient.

Donnons une représentation de ce coefficient :
Prenons un agent au hasard qui, par exemple, a discuté avec trois autres agents. Nous allons regarder si les trois agents ont interagi l'un avec l'autre. Si nous avons seulement, par exemple, deux agents qui ont interagi, cela signifie que nous avons uniquement une connexion qui a été faite sur les trois possibles. Le coefficient de

clustering sera donc égal à $\frac{1}{3}$.

Ce qui nous donne de manière schématique :

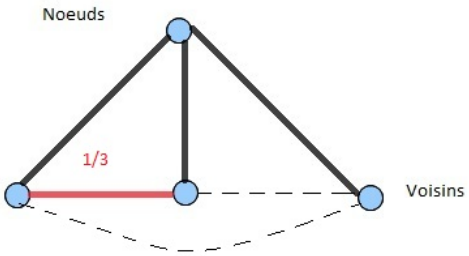


FIGURE 2.8 – Représentation d'un noeuds avec ses trois voisins et les liens qui les unissent. En pointillé, nous avons les liens possibles et en rouge ceux qui existent réellement.

Dans la suite, nous utiliserons le coefficient de clustering moyen.

Le diamètre

Reprenons à nouveau notre modèle sous forme de réseaux. Le diamètre va nous permettre d'avoir plus de renseignements sur notre population.

Donnons une définition du diamètre dans un réseau.

Définition 2.6.2
Le diamètre d'un graphe représente la distance maximale présente entre chaque pair de nœuds.
[4]

Nous aurons comme résultats le diamètre mais la fonction utilisée dans matlab nous donnera également la distance moyenne entre deux nœuds. En sachant que

pour deux nœuds qui ne sont pas reliés la valeur de la distance sera l'infini, pour calculer la distance qui sépare deux nœuds nous utiliserons l'algorithme de Dijkstra. Celui-ci se base sur la méthode du plus court chemin entre deux nœuds.

Cette méthode tient compte du poids de chaque nœud qui sera représenté pour nous par la matrice d'identification I. Pour déterminer la distance entre deux agents, nous allons devoir trouver le plus court chemin entre ceux-ci et cela se fait en additionnant le poids des arêtes entre les deux.

C'est pourquoi nous devrons utiliser l'algorithme de Dijkstra qui est donné par wikipedia d'une manière simplifiée mais assez claire :

Algorithme 2.6.1 (Algorithme de Dijkstra)

```
Dijkstra(noeuds, fils, distance, debut, fin)
  Pour n parcourant noeuds
    n.parcouru := infini   // Peut être implémenté avec -1
    n.precedent := 0
  Fin pour
  debut.parcouru := 0
  DejaVu := liste vide
  PasEncoreVu := noeuds
  Tant que PasEncoreVu != liste vide
    n1 := minimum(PasEncoreVu)   // Le noeud dans PasEncoreVu avec parcouru le plus
    DejaVu.ajouter(n1)
    PasEncoreVu.enlever(n1)
    Pour n2 parcourant fils(n1)   // Les noeuds reliés à n1 par un arc
      Si n2.parcouru > n1.parcouru + distance(n1, n2)   // distance correspond au p
        n2.parcouru := n1.parcouru + distance(n1, n2)
        n2.precedent := n1   // Dis que pour aller à n2, il faut passer par n1
      Fin si
    Fin pour
  Fin tant que
```

```
  chemin := liste vide
  n := fin
  Tant que n != debut
    chemin.ajouterAvant(n)
    n := n.precedent
  Fin tant que
Renvoyer chemin
```

[5]

Prenons un exemple :

Nous avons 6 agents après application de notre modèle et nous aimerions calculer la distance entre l'agent 1 et l'agent 6. Mais également le diamètre de notre population.

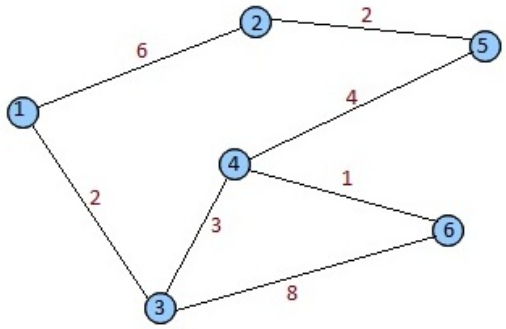


FIGURE 2.9 – Représentation des six agents sortant de la simulation avec les liens qui les unissent et le poids de ceux-ci.

Nous allons donc prendre le plus court chemin entre l'agent 1 et l'agent 6. Il existe plusieurs chemins possibles :

1. $[1, 3, 6]$ dont le poids est de 10.
2. $[1, 3, 4, 6]$ dont le poids est de 6.

3. $[1, 2, 5, 4, 6]$ dont le poids est de 13.
4. $[1, 2, 5, 4, 3, 6]$ dont le poids est de 23.

Nous choisirons donc le deuxième chemin.

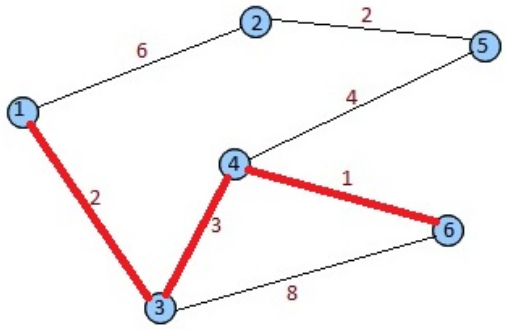


FIGURE 2.10 – Représentation du chemin le plus court entre l'agent 1 et l'agent 6.

En ce qui concerne le diamètre, dans notre population il est donc égal à la distance maximale entre chaque paire de points, or nous avons les distances suivantes entre chaque point :

Nœuds	1	2	3	4	5	6
1	0	6	2	5	8	6
2	6	0	8	6	2	7
3	2	8	0	3	7	4
4	5	6	3	0	4	1
5	8	2	7	4	0	5
6	6	7	4	1	5	0

Nous trouvons donc un diamètre de 8 et pour la distance moyenne, nous obtenons :

$$\begin{aligned} dist &= \frac{6+2+5+8+6+8+6+2+7+3+7+4+4+1+5}{15} \\ &= 4.93333 \end{aligned}$$

Analyse

Revenons maintenant à notre algorithme et analysons les différents coefficients de clustering moyens, les diamètres et les distances moyennes suivant les $d_I = 1 \cdots 5$ afin d'affiner notre analyse. Nous conserverons une population de 1000 agents décrits par 20-bits. Nous allons également représenter graphiquement notre matrice de coefficients de clustering afin de pouvoir aller plus loin dans notre analyse, en sachant que cette matrice nous donne les coefficients de clustering de chaque agent. Nous allons représenter celle-ci en reprenant pour chaque d_I la fréquence cumulée avec laquelle apparaisse ces coefficients.

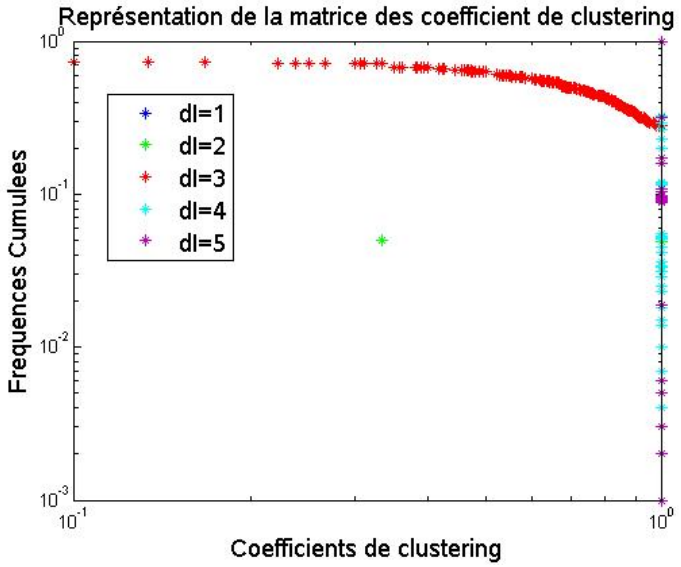


FIGURE 2.11 – Représentation de la fréquence cumulée des coefficients de clustering

Afin d'obtenir les résultats pour le coefficient de clustering, le diamètre et la

distance moyenne, nous avons utilisé des fonctions créées en matlab, nous ressortant uniquement ces trois éléments. Ce qui nous donne ce tableau de résultats que nous analyserons juste ci-dessous en combinaison avec notre graphique d'interaction :

	$d_I = 1$	$d_I = 2$	$d_I = 3$	$d_I = 4$	$d_I = 5$
Coefficient de clustering moyen	0	0.04933	0.57577	0.9992	0.9995
Diamètre	inf	inf	inf	8	7
Distance moyenne	inf	inf	inf	3.65971	3.98871

FIGURE 2.12 – Tableau donnant les résultats du coefficient de clustering moyen, du diamètre et de la distance moyenne sur une simulation avec $k = 20$, $N = 1000$ suivant $d_I = 1 \cdots 5$

$d_I = 1$

Dans ce premier cas, nous avons un coefficient de clustering moyen qui est de 0. Ce qui signifie que soit nous avons un agent qui communique avec plusieurs autres agents, mais que ces derniers ne communiquent pas entre eux, soit cet agent ne parle qu'à une seule personne, il n'y a donc pas de lien possible.

En effet, la matrice des coefficients nous montre que pour $d_I = 1$, il n'y a pas de fréquence du tout. Tous nos coefficient sont donc égaux à 0.

En ce qui concerne le diamètre et la distance moyenne, ils sont infinis ce qui nous permet de dire qu'au moins un des agents n'a aucun lien de communication et donc aucun ami qui se connaissent entre eux. Nous pouvons expliquer ce phénomène à nouveau par le fait que notre d_I est trop petit pour qu'il y ait une bonne communication entre les agents. Ce qui confirme le fait que nous obtenons une population très fragmentée dans les résultats précédents.

$d_I = 2$

Pour un $d_I = 2$, nous avons un coefficient de clustering moyen de 0.0493 ce qui correspond plus ou moins à $\frac{1}{20}$. Cela signifie que sur les 20 agents avec qui un agent a eu des contacts, nous aurons seulement une connexion entre deux. Ce qui est assez faible.

En effet, sur la matrice des coeficients, nous pouvons voir que la communication entre les agents a augmenté puisque nous avons à présent un point qui est représenté. Cependant, cela reste très faible avec une fréquence inférieur à 0.1.

En ce qui concerne le diamètre et la distance moyenne, ils sont toujours infinis. Nous avons donc une amélioration mais qui n'est pas assez forte que pour avoir vraiment de bonnes connexions entre nos agents. Par ailleurs, le diamètre vaut l'infini ce qui signifie qu'il y a dans nos agents, deux d'entre eux qui ne sont pas du tout connectés. Mais cela signifie également que pour faire la moyenne de nos distances, nous avons un infini qui est à prendre en compte et qui met la moyenne automatiquement à l'infini.

$d_I = 3$

Prenons maintenant $d_I = 3$, nous obtenons un coefficient de clustering moyen de 0.576, ce qui nous donne entre $\frac{1}{2}$ et $\frac{2}{3}$ des liens qui seront formés entre les agents voisins d'un même agent. Ceci est vraiment plus élevé que dans la situation précédente. Nous pouvons donc nous dire qu'il y a eu beaucoup de communications, ce qui a été facilité par une augmentation du seuil.

Regardons à présent notre matrice des coefficients, nous remarquons qu'il y a plus de 70% des agents qui ont un coefficient supérieur ou égal à 1. Il est donc normal d'avoir beaucoup plus de communications.

Malheureusement, nous avons une nouvelle fois un diamètre infini, ce qui nous donne donc une distance moyenne infinie. Mais à nouveau, l'analyse de la matrice d'interaction peut nous être utile afin d'observer si nous sommes dans un cas où il y

a beaucoup d'agents qui ne sont pas en communication, ou si nous avons seulement un petit nombre d'agents qui ne le sont pas.

$d_I = 4$

Analysons maintenant les résultats obtenus avec un $d_I = 4$. Le coefficient de clustering moyen est très élevé puisqu'il est de 0.9993. Ceci nous permet de dire que nous nous trouvons presque dans le cas où "tous les amis des mes amis sont mes amis". Ceci conforte les résultats obtenus précédemment qui nous montraient une très grande communication lorsque le d_I était supérieur à 3.

Ces résultats se retrouvent également dans notre matrice puisque l'on peut voir que la plupart des coefficient se trouvent en 1. Ceci confirme la grande communication entre les agents.

Maintenant, nous savons qu'il est toujours possible de trouver un chemin entre deux agents, car le diamètre est de 8 et la distance moyenne est de 3.660. On peut voir que les chemins sont assez courts avec un poids plutôt faible. Observons notre matrice d'interaction pour en savoir plus.

$d_I = 5$

Passons maintenant à notre dernier seuil $d_I = 5$, nous nous retrouvons dans une situation de population totalement en contact. En effet, le coefficient de clustering moyen est de 0.9996, ce qui signifie que tous les agents qui ont été en contact avec un autre agent le sont également entre eux sans exception.

Nous pouvons voir également avec notre matrice que nous avons atteint la fréquence cumulée de 1. Nous avons donc que tous nos agents ont des contacts qui ont des contacts entre eux. Et nous pouvons même dire beaucoup de contacts puisque nous nous trouvons très proche d'un coefficient égal à 1.

En ce qui concerne le diamètre, il est de 7 et la disatnce moyenne de 3.989. Nous nous retrouvons dans le même cas que pour $d_I = 4$ avec beaucoup de commu-

nications entre les différents agents et des agents qui ont tous des voisins communs ce qui permet de trouver un chemin entre tous.

En effet, nous obtenons des résultats très similaires à ceux obtenus avec $d_I = 4$, où chaque individu a au moins communiqué une fois avec tous les agents. Il nous est donc possible de trouver un chemin entre chaque agent. Nous avons donc plus de facilités à communiquer et plus de facilités à se rapprocher l'un de l'autre. C'est pourquoi nous avons un consensus.

Dans le chapitre suivant, nous appliquerons l'algorithme qui a été mis en place sur un système d'élection. Pour cela, nous utiliserons toute l'analyse qui a été faite précédemment.

Chapitre 3

Élections

3.1 Introduction

Dans le chapitre précédent nous avons mis en place un algorithme qui nous permettait de faire interagir une population en fonction du nombre d'opinions qu'elle partageait. Nous nous sommes basés sur le fait que deux individus doivent être assez proches l'un de l'autre afin de pouvoir interagir. Nous avons ensuite analyser cette algorithme de manière à pouvoir déterminer quel seuil était approprié dans certaines situations.

Nous allons maintenant nous concentrer sur un seuil en particulier. En effet, nous aimerions à présent utiliser notre algorithme dans un cas spécifique, celui des élections. C'est pourquoi nous allons utiliser un seuil qui nous permet d'arriver au terme de notre simulation à un consensus total ou quasi-total. Nous allons donc maintenant travailler $d_I = 5$.

3.2 Algorithme pour les élections

Afin de réaliser cette application aux élections, nous allons procéder par étape pour avoir plus de clarté. Nous avons pu constater dans le chapitre précédent les différents résultats obtenus lors de l'utilisation de seuils différents variant de 1 à 5. Dans notre cas des élections, le but étant d'arriver à un éventuel consensus, nous choisirons de travailler avec $k = 20$ puisque toute l'analyse a été faite avec ce

nombre de bit là et pour atteindre cette éventuel consensus nous choisirons $d_I = 5$.

Donnons les étapes de notre nouvel algorithme :

1. Création de Q candidats différents et de k-bits de manière aléatoire.

2. Réalisation de l'algorithme sur une population de N agents de k-bits.
   (a) création de manière aléatoire d'une population de N agents possédant k-bits.
   (b) un processus dynamique répété N fois :
      a. choix aléatoire de deux agents c_i et c_j.
      b. calcul de la distance de hamming. Si $h(c_6, c_8) \leq d_I$, alors un des bits inégal de c_i choisi aléatoirement est changé.
   (c) calcul de la distance de hamming pour chaque agent avec tous les Q candidats afin de déterminer avec lequel il est le plus proche.
   (d) création d'un tableau de vote au cours du temps.
   (e) Fin du processus lorsqu'il n'y a plus d'échange possible.

Après la réalisation de notre algorithme nous obtenons un tableau regroupant les différents votes reçus par les Q candidats au cours de l'algorithme.

3.3 Distribution de votes

Nous allons maintenant mettre en pratique ce qui a été expliqué précédemment en prenant $N = 1000$, $d_I = 5$ et $Q = 5$, ce qui nous donne les résultats suivants.

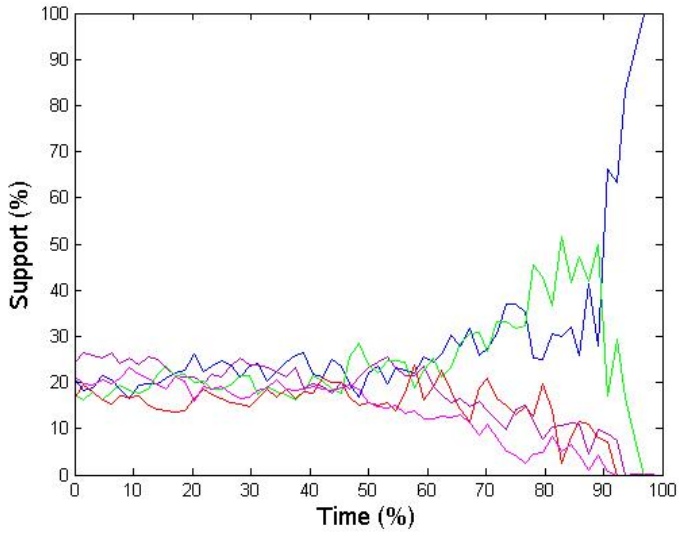


FIGURE 3.1 – Représentation de l'évolution dans le temps du nombre de votes reçu par chaque candidat.

Nous pouvons voir différents comportements durant l'élection.

Premièrement, nous avons la "burn-in-phase", c'est à dire le début de notre simulation là où les candidats restent sur le même pied d'égalité. Dans notre cas, cela correspond aux résultats entre 0 et 60% du temps. Nos candidats restent aux coudes à coudes avec des votes qui vont de 10 à 30%.

Deuxièmement, nous avons la "transient-phase" qui correspond à l'état entre la fragmentation et le consensus c'est-à-dire que certains candidats commencent à se détacher des autres. Pour nous, cette phase se situe entre 60 et 90%. Nous pouvons voir deux candidats prenant l'avantage sur les trois autres.

Et enfin troisièmement, nous avons la "final dynamic", c'est-à-dire la partie après 90% où nous avons un candidat qui s'échappe et qui prend le dessus sur ses concurrents pour finalement prendre les votes de toute la population.

Bien sûr dans un cas réel, nous n'aurions jamais un candidat qui gagne des élections avec 100% des votes, c'est pourquoi lorsque nous voudrons comparer nos

résultats aux résultats d'élections réels, nous prendrons ceux obtenus en transient-phase, c'est-à-dire les résultats obtenus la plupart du temps entre 70 et 95%.

Maintenant, nous allons découper l'évolution de notre élection en petite fenêtres de temps, afin de voir comment se comporte la distribution des votes reçus. Nous allons donc découper notre élection en fenêtres de 20% de temps, ceci nous donne les résultats suivants :

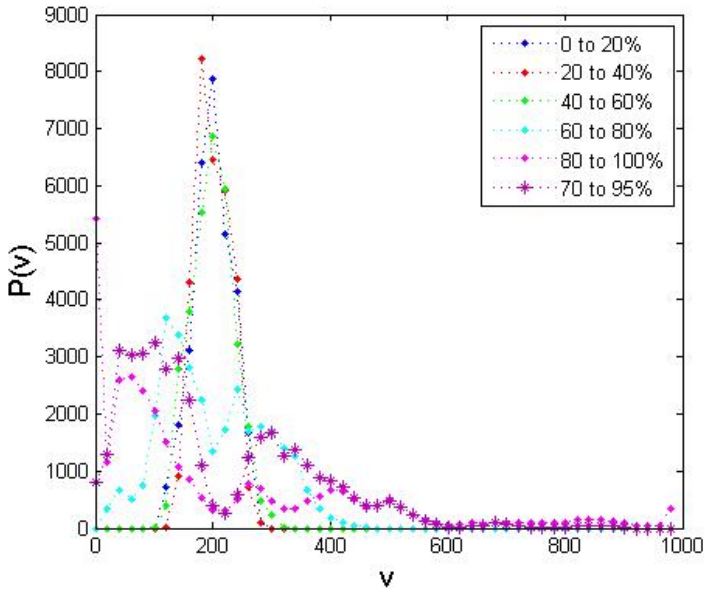


FIGURE 3.2 – Représentation de la distribution des votes reçus lors de la simulation d'élection

Nous pouvons voir que durant la "burn-in-phase", c'est-à-dire de 0 à 60%, notre distribution suit une loi normale avec un $\mu = 200$ qui correspond à $\frac{N}{Q}$.

Ensuite nous pouvons voir qu'entre 60 et 80% la distribution semble plus harmonieuse. Pour enfin terminer l'élection avec un grand nombre de votes à 0. Les deux cas extrêmes n'étant pas très réalistes, cela nous confirme que nous devons regarder ce qu'il y a entre, c'est-à-dire la "transient-phase" qui comme nous pouvons le voir se situe entre ces deux états sur notre graphique.

3.4 Comparaison avec des données réelles

Nous allons à présent comparer les données obtenues grâce à l'algorithme suivant différentes populations mais aussi différents candidats, à des élections qui se sont déroulées en 2003 en Finlande.

Pour cela, nous avons fait tourner notre algorithme pour des populations de $N = 1000$, 2000 et 4000 agents avec un nombre de 5 ou 10 candidats.

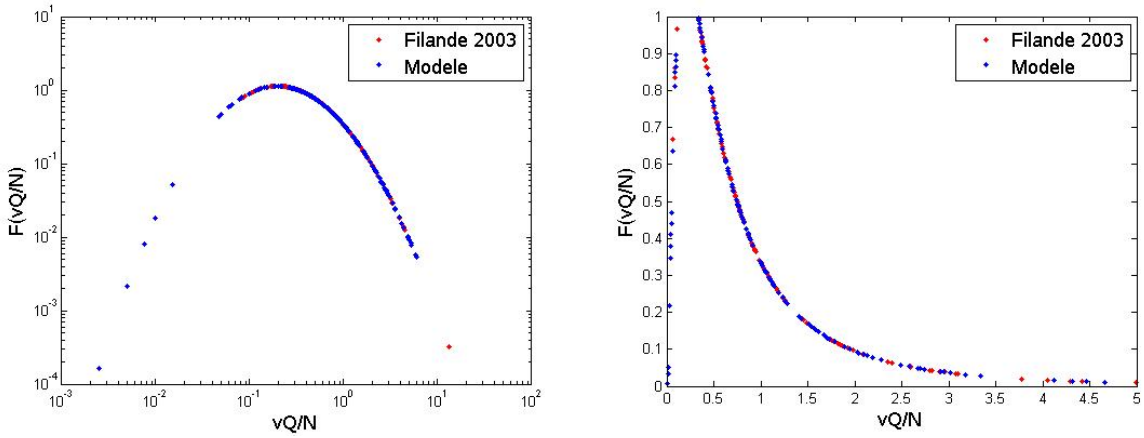
Ensuite, nous avons représenté graphiquement les résultats obtenus, en gardant uniquement les périodes entre 70% et 95% des élections. Puisqu'elle représente la situation la plus réaliste.

Nous allons également utiliser notre fonction de ré-échelonnement :

$$F\left(\frac{vQ}{N}\right) = \frac{N}{\sqrt{2\pi}\sigma vQ} \mathrm{e}^{-\frac{\left(\log\left(\frac{vQ}{N}\right)-\mu\right)^2}{2\sigma^2}} \tag{3.1}$$

$\mu = 0.54$ et $\sigma^2 = -2\mu$. [2]
afin de pouvoir comparer nos résultats comme expliqué dans le chapitre 1.

Pour plus de clarté, nous avons pris un ensemble de 170 données afin de ne pas alourdir le graphique, ceci nous donne les résultats suivants :



(a) Représentation sous forme loglog

(b) Représentation sous forme "normale"

FIGURE 3.3 – Comparaison entre les résultats électoraux de la Finlande et notre modèle entre 70% et 95% d'exécution

Nous pouvons observer une similitude entre les élections de la Finlande de 2003 et notre modèle, ce qui nous démontre qu'il existe bien une loi qui permet d'avoir les mêmes propriétés entre notre modèle et des élections réelles.

Chapitre 4

Modèle ouvert

4.1 Introduction

Nous allons nous intéresser à présent à notre algorithme de manière encore plus réaliste. En effet, dans la vie de tous les jours la population n'est pas fixe, elle varie par les naissances et les décès qu'il peut y avoir, mais aussi elle varie par l'immigration et l'émigration. Ce qui nous permet d'avoir une population beaucoup plus réaliste et donc des résultats plus proches de la réalité.

Ce chapitre sera donc basé sur la réalisation de l'algorithme appliqué dans ces conditions, suivi d'une analyse.

4.2 Adaptation de l'algorithme

Afin de réaliser notre adaptation, nous devons tenir compte de deux nouveaux paramètres T et p. Le paramètre T représente la durée de temps après laquelle nous devons effectuer une sortie et une entrée d'agents, nous avons choisi de donner à notre paramètre les valeurs suivantes : 100, 500, 1000 et 5000.
Celles-ci correspondent aux nombres de fois où l'on effectue une itération de l'algorithme. Donc pour le premier T, il y a des agents qui rentrent et qui sortent toutes les 100 itérations.

Maintenant expliquons le paramètre p, celui-ci correspond au nombre de personnes qui sortent et qui rentrent dans la population. Celui-ci prendra comme valeurs 0.5%, 1%, 5% et 10% de la population.

4.3 Analyse

Nous allons maintenant appliquer notre algorithme avec une simulation unique sur une population aléatoire où $N = 1000$, $k = 20$ et $d_I = 5$. Pour les T et p repris ci-dessus, c'est-à-dire : $T = [100, 500, 1000, 5000]$ et $p = [5, 10, 50, 100]$. Afin de ne pas avoir des algorithmes qui tournent indéfiniment, nous avons mis un nombre d'itérations maximales de 35000. Nous obtenons les résultats suivants :

$T = 100$

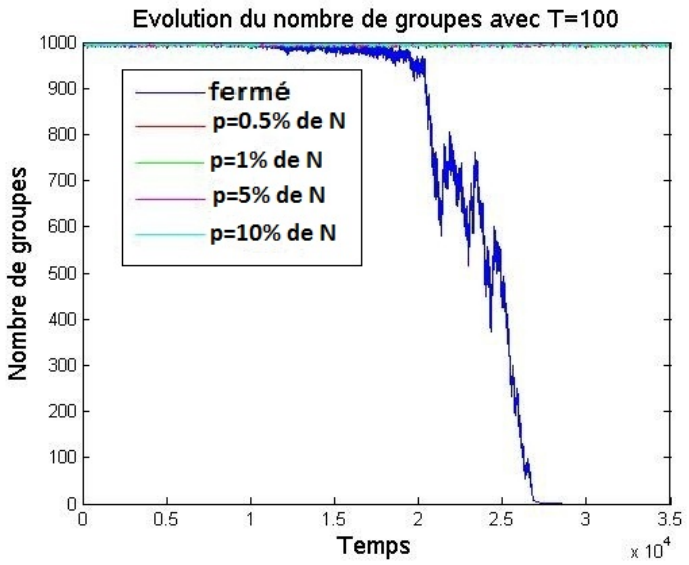


FIGURE 4.1 – Évolution du nombre de groupes pour $T = 100$ avec les différents p possibles et le modèle ouvert

Le T étant assez petit, nous allons avoir beaucoup de mouvements dans notre population et donc les groupes seront plus élevés puisque les agents auront plus

de difficultés à se rapprocher l'un de l'autre.

Nous pouvons donc voir que notre algorithme avec du bruit n'arrive pas à faire descendre le nombre de groupes que ce soit pour $p = 5, 10{,}50$ ou encore $p = 100$. Nous restons continuellement avec un nombre de groupes proches de 1000, c'est à dire que presque tous nos agents sont différents et ne partagent donc pas les mêmes idées.

$T = 500$

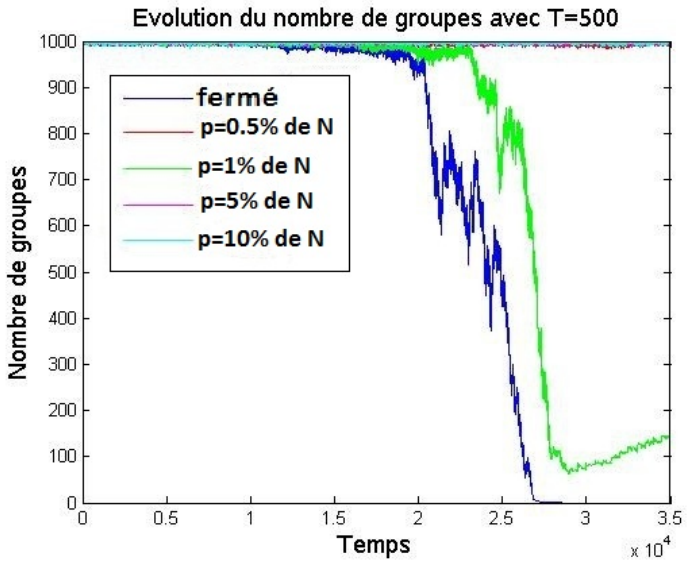


FIGURE 4.2 – Évolution du nombre de groupes pour $T = 500$ avec les différents p possibles et le modèle ouvert

Nous pouvons constater une certaine évolution dans ce cas-ci. Nous pouvons voir que dans le cas où $p = 10$ le nombre de groupes se comporte de la même manière que le modèle fermé jusqu'à un certain point. Effectivement, nous pouvons voir que lors de l'entrée de nouveaux éléments et la sortie d'anciens vers 30000 itérations créent une remontée du nombre de groupes. Cependant, nous pouvons constater que pour $p = 5$ nous gardons un nombre de groupes proches de 1000. Les connexions ont donc été meilleures pour $p = 10$.

Ce résultat étant assez surprenant, nous pouvons nous dire que cela doit sûrement s'expliquer à cause des entrées et des sorties des agents. Celles-ci ont certainement forcé la population à rester fragmentée et nos itérations étant limitées, nous obtenons ce type de résultats. Malheureusement, il fallait bien arrêter les itérations à un moment donné au risque de perdre un peu de qualité. Mais nous permettant de ne pas avoir des algorithme infini.

Nous constatons également qu'il n'y a toujours pas de changement pour $p = 50, 100$, le nombre de groupes reste proche de 1000.

Le changement reste trop rapide, les agents n'ont pas le temps d'échanger assez d'idées que pour former de gros groupes dont la suppression de certains individus et la venue de nouveaux ne bouleverseraient pas.

$T = 1000$

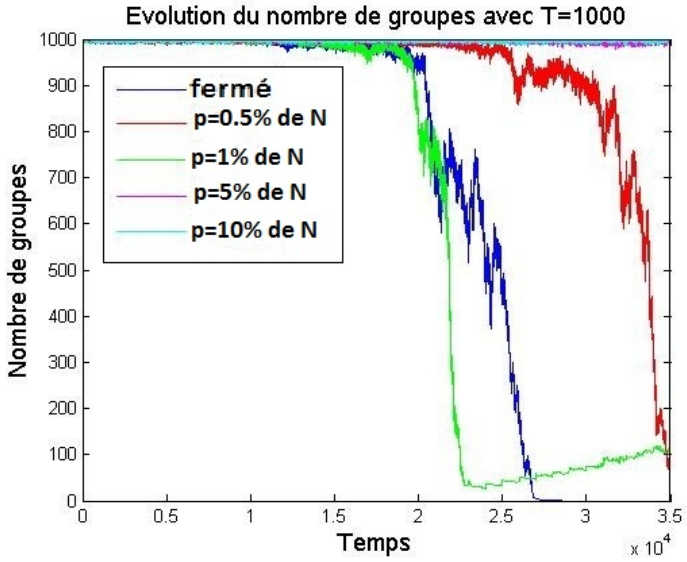


FIGURE 4.3 – Évolution du nombre de groupes pour $T = 1000$ avec les différents p possibles et le modèle ouvert

Premièrement, on constate que $p = 10$ va être légèrement plus rapide que le modèle fermé dans sa convergence. A nouveau, nous observons une remontée du

nombre de groupes entre 20000 et 25000 itérations, causée par l'entrée et la sortie d'agents.

Ensuite, nous constatons que pour $p = 5$ nous avons également une convergence vers un nombre de groupe égal à 1. Celle-ci se faisant plus tardivement.

Et enfin, nous pouvons également voir que pour $p = 50$ et $p = 100$, il n'y a toujours pas de changement le nombre de groupes reste à 1000.

$T = 5000$

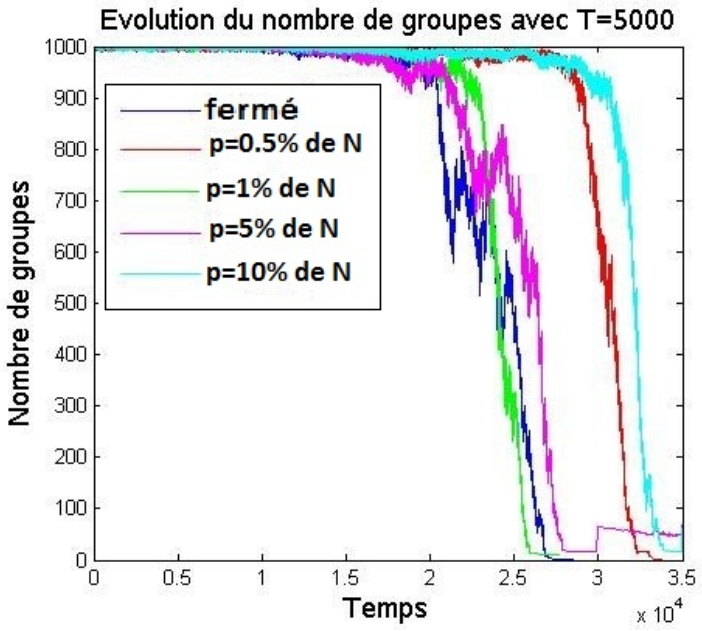


FIGURE 4.4 – Évolution du nombre de groupes pour $T = 5000$ avec les différents p possibles et le modèle ouvert

Le T dans ce cas-ci étant vraiment élevé, il se rapproche beaucoup du modèle fermé. Nous pouvons donc observer que pour $p = 5, 10, 50$ et même 100 le modèle converge vers un nombre de groupes égal à 1 comme le modèle fermé. On peut malgré tout voir pour $p = 50$ une légère remontée du nombre de groupe.

Nous avons donc pu voir qu'avec un $p = 10\%$ de N, il est assez improbable de voir un candidat s'échapper du reste du groupe. Ce taux est beaucoup trop élevé. Il faut donc rester dans des valeurs plus plausibles.

4.4 Application aux élections

Lors de la section précédente, nous avons uniquement tenu compte de l'évolution du nombre de groupes lors de la simulation pour $T = [100, 500, 1000, 5000]$ et $p = [5, 10, 50, 100]$. Maintenant, nous allons tenir également compte de leur choix de candidat durant la simulation. Nous avons donc utilisé exactement la même simulation que précédemment mais en l'appliquant aux élections.

Nous allons à présent regarder les résultats obtenus.

$T = 100$

Rappelons que dans ce premier cas, lors de l'évolution de groupe pour aucun p nous n'avions de convergence.

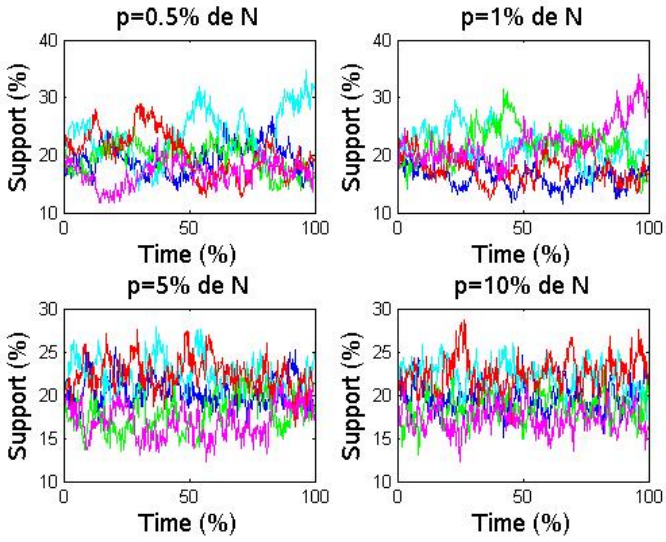


FIGURE 4.5 – Élections dans un modèle ouvert avec $T = 100$ et $p = 5, 10{,}50$ et 100

$p = 0.5\%$ **de N**

Nous commençons pour ce premier p, avec un temps très court entre les différentes entrées et sorties des agents. Cependant, le nombre de personnes en échange s'élevant seulement à 5, nous pouvons voir un candidat qui tente de s'échapper quelque peu des autres. Sans toute fois que ce soit réellement significatif, les votes étant au maximum de 30%.

$p = 1\%$ **de N**

Nous pouvons voir dans ce deuxième cas, à nouveau un des candidats qui semble vouloir s'échapper des quatre autres. Mais nous ne pouvons toujours pas voir ce seul candidat se détacher considérablement et cela confirme le fait que le nombre de groupes n'évolue pas et que la population reste totalement fragmentée.

$p = 5\%$ **de N**

Dans ce troisième cas, nous puvons voir toues les candidats très proches l'un de l'autre. Aucun d'eux ne semble pouvoir s'échapper. En effet, nous restons avec des votes qui sont en-dessous des 30% durant toute la simulation.

$p = 10\%$ **de N**

Dans ce quatrième cas, le pourcentage de personnes qui entrent et qui sortent sur un laps de temps assez court est trop élevé ce qui crée une situation où à nouveau, nos 5 candidats sont au coude à coude et ne semblent pas pouvoir se détacher.

Essayons maintenant d'augmenter les temps entre l'entrée et la sortie d'agents, afin de voir si notre modèle en est moins affecté.

$T = 500$

Lors de cette simulation, nous avions uniquement $p = 10$ qui montrait une convergence mais avec une remontée du nombre de groupes vers la fin.

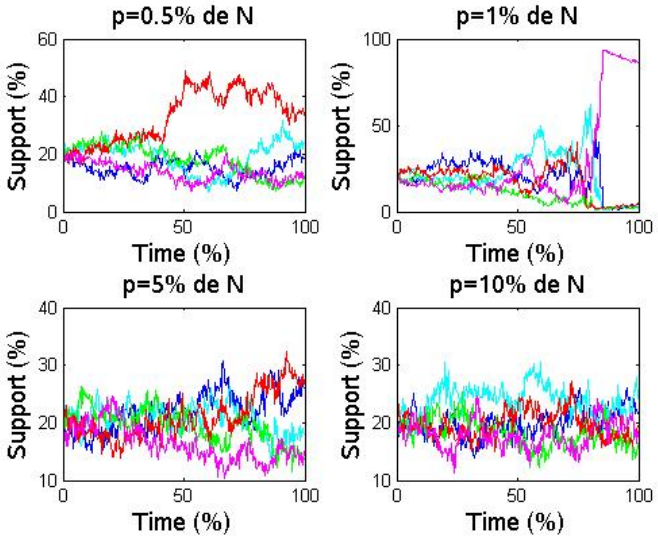


FIGURE 4.6 – Élections dans un modèle ouvert avec $T = 500$ et $p = 5, 10, 50$ et 100

$p = 0.5\%$ de N

Dans ce premier cas, on constate qu'un candidat tente de se détacher légèrement des autres mais qui finalement voit son nombre de votes diminuer pour se retrouver presqu'au même niveau que les quatre autres. Ceci étant du à une nouvelle vague d'entrées et de sorties d'agents.

$p = 1\%$ de N

Le temps entre les différentes entrées et sorties ayant été allongé, nous pouvons constater que pour $p = 10$ nous avons un candidat qui s'échappe. Bien sûr, nous pouvons voir qu'au terme de la simulation son nombre de voix diminue légèrement ce qui vient confirmer les résultats obtenus lors de l'évolution du

nombre de groupes.

De plus, nous pouvons voir qu'il était en concurrence avec deux autres candidats, les deux autres ayant moins de votes, mais il a tout de même réussi à s'échapper malgré les changements opérés dans la population. Ceci confirme les résultats obtenus avec le nombre de groupes qui tendait vers un seul et unique groupe et donc pour des votes allant vers un seul candidat.

$p = 5\%$ **de N**

Dans le cas présent, nous pouvons voir deux groupes se former avec un premier regroupant deux agents regroupant chacun un peu plus de 25% des votes et un deuxième groupe de trois candidats ayant chacun moins de 20% des votes.

$p = 10\%$ **de N**

On peut voir dans ce cas-ci beaucoup de mouvements entre les différents candidats ce qui est provoqué par un trop grand nombre d'agents qui entrent et qui sortent. C'est pourquoi nous avons des candidats qui restent toujours en-dessous des 30% et qui ne peuvent pas se détacher des autres.

$T = 1000$

Lors de l'évolution du nombre de groupes, nous avions $p = 5$ et $p = 10$ qui montraient une convergence vers un nombre de groupes égal à 1.

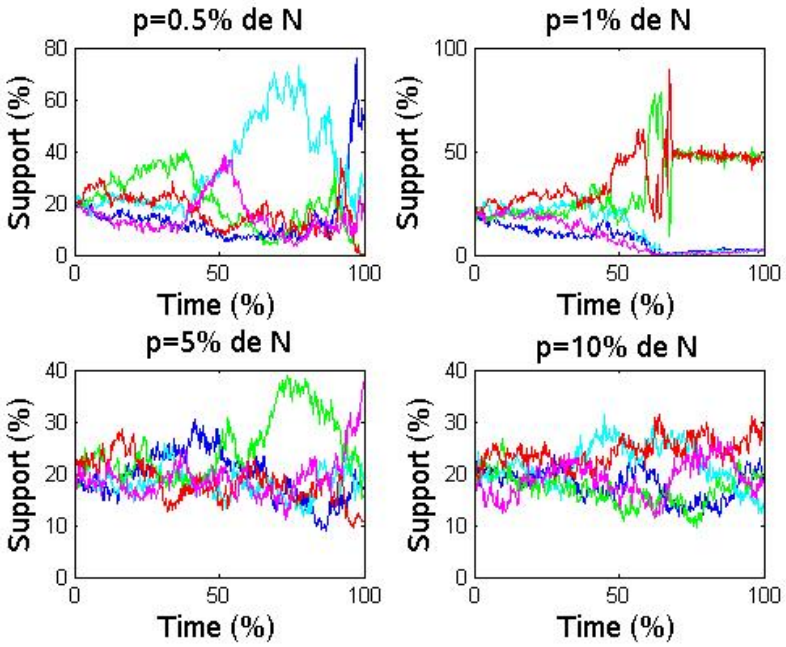


FIGURE 4.7 – Élections dans un modèle ouvert avec $T = 1000$ et $p = 5, 10, 50$ et 100

$p = 0.5\%$ de N

Dans ce premier cas, nous nous avons un candidat qui s'échappe vers la fin de la simulation en laissant les quatre autres derrière lui. Mais on peut voir qu'après 95% de simulation il redescend un petit peu au profit d'un autre candidat.

$p = 1\%$ de N

Dans ce deuxième cas, nous obtenons une situation un petit peu particulière avec deux candidats qui ne savent absolument pas se détacher l'un l'autre jusqu'au terme de la simulation. Et qui regroupe chacun plus ou moins 50% des votes ce qui ne laisse aucune chance aux trois autres candidats.

$p = 5\%$ de N

Dans ce troisième cas, on peut voir un candidat qui tente de s'échapper quelque peu vers la fin. Cependant, il ne récolte à ce moment-là que moins de 40% des votes.

Les quatre autres candidats se retrouvant en dessous des 25% des votes.

$p = 10\%$ de N

Dans ce dernier cas, on peut voir quatre candidats au coude à coude au niveau des 20% de votes, alors que le dernier candidat se trouve légèrement au-dessus.

$T = 5000$

Ce dernier cas, nous montrait lors de l'évolution du nombre de groupes que pour tous les taux, nous avions une convergence.

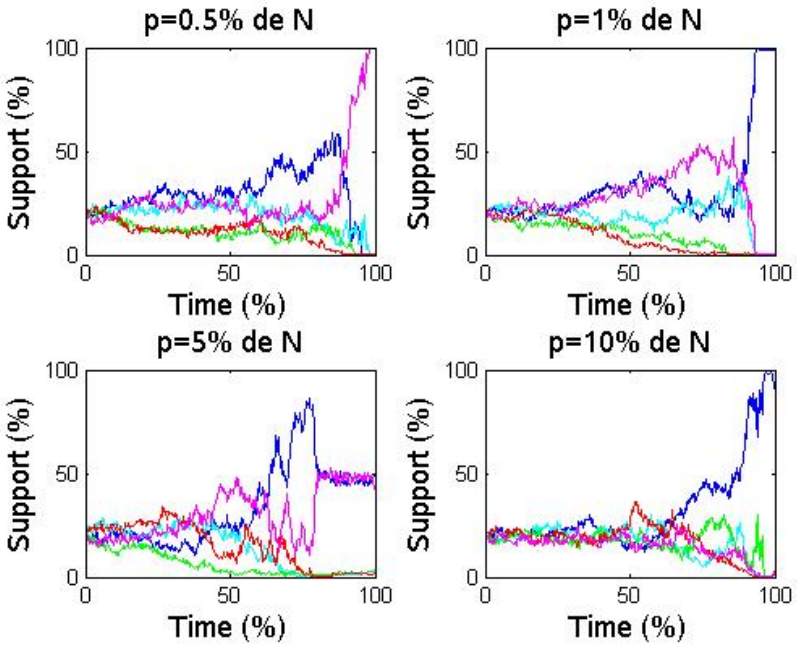


FIGURE 4.8 – Élections dans un modèle ouvert avec $T = 5000$ et $p = 5, 10, 50$ et 100

Les différents cas étant assez similaires, nous les développerons en une fois. Effectivement, nous constatons que pour $p = 5, 10$ et 100, nous nous retrouvons

dans une élection de type fermé avec un candidat qui finit par s'échapper au terme de la simulation.

Par contre, nous retrouvons également notre cas où nous avons deux candidats qui se partagent le nombre de votes et ne parviennent pas à se détacher. Laissant les trois autres candidats avec aucun vote.

Nous pouvons donc dire en voyant les différents résultats obtenus que pour $p = 100$, excepté pour notre T très élevé ($T = 5000$), nous aurons toujours des candidats au coude à coude qui auront peu de chance de voir un d'eux s'échapper. Si l'on veut voir un candidat il faut se placer avec un p plus petit mais aussi un T pas trop petit, car avec un T trop faible nous nous retrouvons avec des échanges trop rapides qui ne permettent pas une bonne communication entre les agents et qui ne permettent donc pas au candidat de vraiment se détacher des autres.

4.5 Comparaison avec des données réelles

Nous allons à présent comparer les données obtenues grâce à l'algorithme de notre modèle ouvert suivant les différentes situations décrites plus haut.

Nous avons décidé d'utiliser $p = 0.5\%$ et $p = 1\%$ avec $T = 1000$ et 5000. Pour cela, nous avons gardé les résultats obtenus précédemment avec $N = 1000$, $k = 20$ et nos 5 candidats.

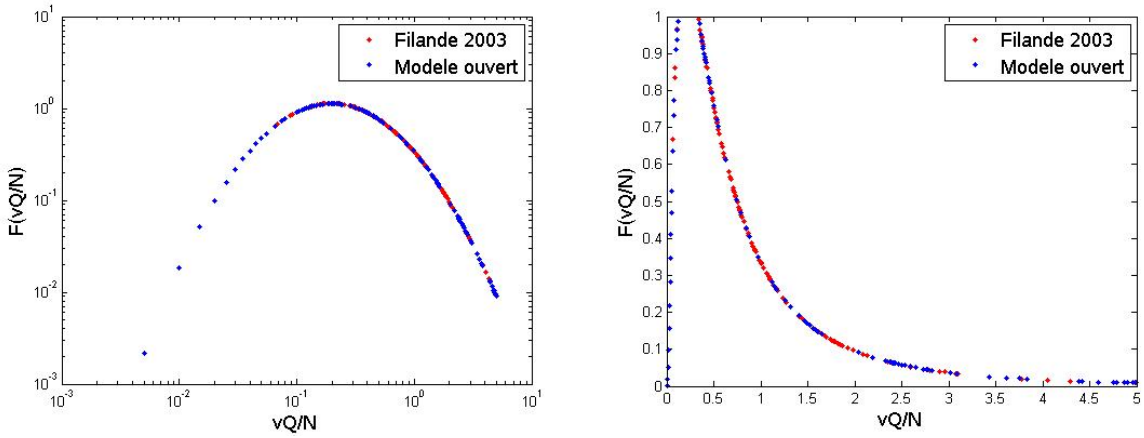
Ensuite, nous avons à nouveau représenté graphiquement les résultats obtenus, en gardant uniquement les périodes entre 70% et 95% des élections. Puisqu'elles représentent la situation la plus réaliste. Comme nous avions fait lors du modèle fermé.

Nous allons également utiliser notre fonction de ré-échelonnement :

$$F\left(\frac{vQ}{N}\right) = \frac{N}{\sqrt{2\pi}\sigma vQ} \mathrm{e}^{-\frac{\left(\log\left(\frac{vQ}{N}\right)-\mu\right)^2}{2\sigma^2}} \tag{4.1}$$

$\mu = 0.54$ et $\sigma^2 = -2\mu$. [2]

Pour plus de clarté, nous avons pris à nouveau un ensemble de 170 données afin de ne pas alourdir le graphique, ceci nous donne les résultats suivants :



(a) Représentation sous forme loglog

(b) Représentation sous forme "normale"

FIGURE 4.9 – Comparaison entre les résultats électoraux de la Finlande et notre modèle ouvert entre 70% et 95% d'exécution

On constate que notre modèle ouvert est également similaire aux résultats obtenus lors des élections de Finlande de 2003. Ceci nous informe du bon fonctionnement de notre algorithme, également lorsque nous y mettons du mouvement.

Conclusion

Nous avons pu constater au terme de notre cheminement plusieurs résultats très intéressants.

Premièrement, nous avons d'abord révélé une distribution universelle en nous basant sur les travaux de Fortunato et Castellano ([2]) qui permettait de faire un lien entre les différentes élections de part le monde.

Ensuite, nous avons créé un modèle d'échange à nouveau en nous basant sur des travaux, ceux de Banisch, Araújo et Louçã ([3]). Ce modèle, nous l'avons analysé afin de pouvoir l'adapter à un contexte d'élections. Ceci nous a permis de conforter l'idée de notre distribution universelle qui avait été mise en place. En effet, lorsque nous avons confronté les résultats obtenus avec le modèle et ceux obtenus lors des élections Finlandaises de 2003, nous avons pu constater une correspondance entre les deux.

Finalement, nous sommes sortis de ces différents travaux afin d'aller un petit plus loin. C'est pourquoi nous avons adapté notre algorithme pour nous mettre dans une situation encore plus réaliste en mettant du mouvement dans notre modèle et en le rendant ouvert. Ceci nous a permis de voir notre modèle de manière un peu différente. En effet, les différents résultats qui nous ont été donné montraient que notre modèle réagissait assez rapidement aux différents mouvements d'entrée et de sortie des agents.

Annexes

Algorithme simple

```
function[c,number]=algorithmesimple(dI,N,c)
% function [c,number]=algorithmesimple(dI,N,c)
%
% Cette algorithme a pour but de réaliser l'échange d'opinion
%
%
% Inputs:
% c = population
% N = nombre d'agents
% dI = seuil utilisé pour l'algorithme
% Output:
% number : nombre de groupes durant la simulation
% c : nouvelle population
%
% Moriame Marie
%
%tout l'algo
stop=1;
var=1;
while (stop~=0)
    for m=1:1:N %un pas de l'algo
        condi=0;
        while (condi==0)
```

```
            %choix des deux agents qui entrent en interaction
            i = randi([1,N],2,1);
            condi=ne(i(1),i(2));
        end
        %condition pour voir si les agents echangent des idees ou non
        dist=ne(c(:,i(1)),c(:,i(2)));
[dist IX]=sort(dist,'descend');
 dist=sum(dist);
 place=IX(1:dist);
        if (dist<=dI)&&(dist~=0)
            [t, ~]=size(place);
            j=randi([1,t],1,1);
            c(place(j),i(1))=c(place(j),i(2));
        end

    end

    %On regarde s'il est encore possible de faire interagir deux
    %agents
    ind1=1;
    stop=0;
    C2=sortrows(c');
    C2=C2';
    D=unique(C2', 'rows')';
    [~, col]=size(D);
    number(var)=col;
    while (ind1<col)&&(stop==0)
        ind2=ind1+1;
        while (ind2<=col)&&(stop==0)
            dist=sum(ne(D(:,ind1),D(:,ind2)));
            stop=le(dist,dI);
            ind2=ind2+1;
        end
        ind1=ind1+1;
```

```
    end
    var=var+1;
end
```

Algorithme élections

```
function[c,A,A2,var]=algorithmeelection(dI,N,c,Q,candidat)
% function [c,A,A2,var]=algorithmeelection(dI,N,c,Q,candidat)
%
% Cette algorithme a pour but de réaliser l'échange\\
% d'opinion en s'adaptant aux élections
%
%
% Inputs:
% Q = nombre de candidats
% candidat = Q candidats sous forme k-bits
% c = population
% N = nombre d'agents
% dI = seuil utilisé pour l'algorithme
%
% Output:
% c : nouvelle population
% A : nombre de votes reçus pour chaque candidat
% A2 : nombre de votes reçus pour chaque candidat en %
% var : nombre d'iterations
%
% Moriame Marie
%

%tout l'algo
stop=1;
var=1;
A(var,:)=zeros(1,Q);
% calcul du nombre de votes reçus pour chaque candidat
```

```
for i=1:1:N
    for l=1:1:Q
        %dist(l)=hamming2(candidat(:,l),c(:,i));
        dist(l)=sum(ne(candidat(:,l),c(:,i)));
    end
    [dist IX]=sort(dist);
    egalite=eq(dist,dist(1));
    somme=sum(egalite);
    j=randi([1,somme],1,1);
    A(var,IX(j))=A(var,IX(j))+1;
end
A;
A2(var,:)=A(var,:)./N;
compteur=0;
B=0;
while (stop~=0)&&(compteur<=5000)
    for m=1:1:N %un pas de l'algo
        condi=0;
        while (condi==0)
            %choix des deux agents qui entrent en interaction
            i = randi([1,N],1,2);
            condi=ne(i(1),i(2));
        end
        %condition pour voir si les agents echangent des idees ou non
        %[dist place]=hamming(c(:,i(1)),c(:,i(2)));
        dist=ne(c(:,i(1)),c(:,i(2)));
        [dist IX]=sort(dist,'descend');
        dist=sum(dist);
        place=IX(1:dist);

        if (dist<=dI)&&(dist~=0)
            j=randi([1,dist],1,1);
            c(place(j),i(1))=c(place(j),i(2));
        end
```

```
end

var=var+1;
A(var,:)=zeros(1,Q);
% calcul du nombre de votes reçus pour chaque candidat
for i=1:1:N
    for i=1:1:N
        for l=1:1:Q
            %dist(l)=hamming2(candidat(:,l),c(:,i));
            dist(l)=sum(ne(candidat(:,l),c(:,i)));
        end
        [dist IX]=sort(dist);
        egalite=eq(dist,dist(1));
        somme=sum(egalite);
        j=randi([1,somme],1,1);
        A(var,IX(j))=A(var,IX(j))+1;
    end
    A;
    A2(var,:)=A(var,:)./N;
    %     %On regarde s'il est encore possible de faire interagir deux
    %     %agents
    ind1=1;
    stop=0;
    C2=sortrows(c');
    C2=C2';
    D=unique(C2', 'rows')';
    [~, col]=size(D)
    compteur=compteur+eq(B,col);
    if (compteur==multi)
        compteur;
        multi=multi+100;
    end
    if (col~=B)
```

```
            compteur=0;
            multi=1;
        end
        B=col;
        while (ind1<col)&&(stop==0)
            ind2=ind1+1;
            while (ind2<=col)&&(stop==0)
                %dist=hamming2(D(:,ind1),D(:,ind2));
                dist=sum(ne(D(:,ind1),D(:,ind2)));
                stop=le(dist,dI);
                ind2=ind2+1;
            end
            ind1=ind1+1;
        end
    end
```

Algorithme pour le modèle ouvert

```
function[c,A,A2,var,number]=algobruyantelection(dI,N,c,Q,k,candidat,p,Tinit)
% function[c,A,A2,var,number]=algobruyantelection(dI,N,c,Q,k,candidat,p,Tin
% it)
%
% Cette algorithme a pour but de réaliser l'échange d'opinion en s'adaptant
% aux élections avec bruit
%
%
% Inputs:
% Q = nombre de candidats
% candidat = Q candidats sous forme k-bits
% c = population
% N = nombre d'agents
% dI = seuil utilisé pour l'algorithme
```

```
% k = nombre de bits
% p = taux d'entrée et de sortie d'agents
% T = nombre d'itérations entre chaque entrée et sortie d'agents
%
% Output:
% c : nouvelle population
% A : nombre de votes reçus pour chaque candidat
% A2 : nombre de votes reçus pour chaque candidat en %
% var : nombre d'iterations
% number = nombre de groupes durant la simulation
%
% Moriame Marie
%

%tout l'algo
stop=1;
var=1;
multi=1;
T=Tinit;
A(var,:)=zeros(1,Q);
% calcul du nombre de voix reçus pour chaque candidat
for i=1:1:N
    for l=1:1:Q
        %dist(l)=hamming2(candidat(:,l),c(:,i));
        dist(l)=sum(ne(candidat(:,l),c(:,i)));
    end
    [dist IX]=sort(dist);
    egalite=eq(dist,dist(1));
    somme=sum(egalite);
    j=randi([1,somme],1,1);
    A(var,IX(j))=A(var,IX(j))+1;
end
A;
A2(var,:)=A(var,:)./N;
```

```
compteur=0;
B=0;
while (stop~=0)&&(compteur<=5000)&&(var<=35000)%&&(test1<=N*0.05)
    for m=1:1:N %un pas de l'algo
        condi=0;
        while (condi==0)
            %choix des deux agents qui entrent en interaction
            i = randi([1,N],1,2);
            condi=ne(i(1),i(2));
        end
        %condition pour voir si les agents echangent des idees ou non
        %[dist place]=hamming(c(:,i(1)),c(:,i(2)));
        dist=ne(c(:,i(1)),c(:,i(2)));
        [dist IX]=sort(dist,'descend');
        dist=sum(dist);
        place=IX(1:dist);

        if (dist<=dI)&&(dist~=0)
            j=randi([1,dist],1,1);
            c(place(j),i(1))=c(place(j),i(2));
        end

    end

    var=var+1;
    A(var,:)=zeros(1,Q);
    % calcul du nombre de voix reçus pour chaque candidat
    for i=1:1:N
        for l=1:1:Q
            %dist(l)=hamming2(candidat(:,l),c(:,i));
            dist(l)=sum(ne(candidat(:,l),c(:,i)));
        end
        [dist IX]=sort(dist);
        egalite=eq(dist,dist(1));
```

```
    somme=sum(egalite);
    j=randi([1,somme],1,1);
    A(var,IX(j))=A(var,IX(j))+1;
end
A;
A2(var,:)=A(var,:)./N;
%     %On regarde s'il est encore possible de faire interagir deux
%     %agents
ind1=1;
stop=0;
C2=sortrows(c');
C2=C2';
D=unique(C2', 'rows')';
[~, col]=size(D);
number(var)=col;
compteur=compteur+eq(B,col);
if (compteur==multi)
    compteur;
    multi=multi+100;
end
if (col~=B)
    compteur=0;
    multi=1;
end
B=col;
while (ind1<col)&&(stop==0)
    ind2=ind1+1;
    while (ind2<=col)&&(stop==0)
        %dist=hamming2(D(:,ind1),D(:,ind2));
        dist=sum(ne(D(:,ind1),D(:,ind2)));
        stop=le(dist,dI);
        ind2=ind2+1;
    end
    ind1=ind1+1;
```

```
    end
    % entrée et sortie d'agents
    if (var==T)
        r=randi([1,N],p,1);
        for z=1:1:p
            c(:,r(z))=randi([0,1],k,1);
        end
        T=T+Tinit;
    end
end
```

Bibliographie

[1] Sven Banisch, Tanya Araujo, "On the empirical relevance of the transient in opinion models",http :arxiv.org, 29 mars 2010.

[2] Santo Fortunato and Claudio Castellano, "Scaling and universality in proportional elections", http :arxiv.org, 2007.

[3] Sven Banisch, Tanya Araujo and Jorge Louça, "Opinion dynamics and Communication Networks", http :arxiv.org, 29 avril 2009.

[4] Claudio Castellano, Santo Fortunato and Vittorio Loreto, "Statistical physics of social dynamics", http :arxiv.org, 11 may 2009.

[5] http ://fr.wikipedia.org, 31 mars 2011, 22 avril 2011.

[6] "Election statistics", http :www.stat.fitkhevaalitvaalit2003vaalit2003_vaalitilastot_en.html, 23 avril 2011.