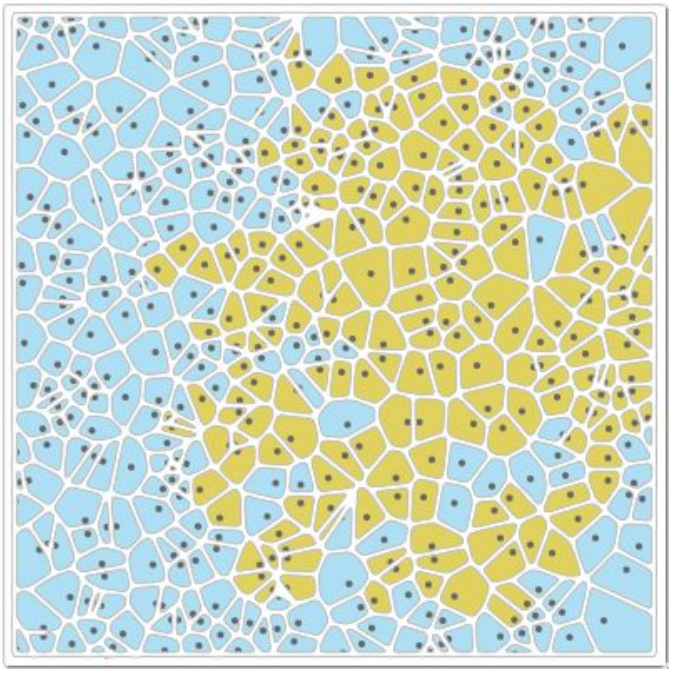
Game Theory for modeling tumor growth

How-Choong Inès GB02

Université de technologie de Compiègne,Compiègne, France

Abstract : Le but de ce projet est de réaliser des recherches bibliographiques afin de répondre aux questions suivantes :
Qu'est ce que la théorie des jeux ? Comment appliquer ce modèle sur la croissance des cellules tumorales ? Existe-t-il des codes en open source ? Sinon, si l'étudiant est intéressé il/elle peut créer et implémenter un code exploratoire (en Matlab ou Python) sous la direction du Pr. Badr Kaoui.

Mot clés: Théorie du jeux, Cellules tumorales,Python



Sommaire

1. Théorie des jeux

La théorie des jeux est un domaine mathématique dont le but est d'étudier les interactions stratégiques entre différents agents. Ce concepte est appliquable dans différents domaines tels que le biologie, la finance, les sciences politiques etc.

La théorie des jeux permet une analyse formelle des problèmes posés par les interactions stratégiques dans un groupe d'agents rationnels poursuivant des buts qui leurs sont propres. Autrement dit, la théorie du jeux est un moyen de modéliser de manière mathématique diffèrentes intéractions.

1. Quelques concepts

Comment est défini un jeu ?

Un jeu est défini par ses limites. La définition impose dans un premier temps la présence d'agents appelés *joueurs.* Ces derniers sont supposés *rationnels* et le but de chacun est de *gagner*.

Afin d'arriver à leur fin, ils doivent réaliser des *stratégies* par *coup*. A chaque issue, les joueurs ont des gains ou des pertes, ce qu'on appelle dans le cadre du jeu des *bénéfices*.

Il existe différents moyens de représenter un jeu : sous forme extensive ou sous forme normale/stratégique.

2. Modélisation

a. Forme normale et stratégique

On peut modéliser un jeu sous forme stratégique grâce à une matrice des bénéfices. Il s'agit d'un tableau à double-entrée qui énumère sur chaque côté les stratégies possibles des joueurs respectifs. Dans la case à la croisée de deux stratégies, on note le couple de gains des deux joueurs.

Donnons un exemple de cadre de jeu :

Soit un jeu Y, ayant deux joueurs qui jouent de manière simultanées. Ces derniers peuvent réaliser deux coups U ou D. Voici les différents gains :

- S'ils adoptent la même stratégie, ils gagnent tous les deux 1 points
- S'ils adoptent des stratégies opposées, ils ne gagnent aucun point.

Voici par exemple, la matrice qu'on peut poser à l'issu de ce jeu:

	Joueur 1		
Joueur 2		U	D
	U	(1,1)	(0,0)
	D	(0,0)	(1,1)

Matrice du jeu Y

Limites de la forme Normale / Stratégique : L'utilisation de la forme normale nécessite un nombre de joueurs fini. De plus, en générale, on fait l'hypothèse que chaque joueur choisit sa stratégie sans savoir la stratégie de l'autre.

b. Forme extensive

La forme extensive est défini par un graphe de type arbre.

Dans tous les jeux, les décisions peuvent être représentées par un arbre, dont chaque nœud est associé au joueur qui décide. Chaque option constitue une branche. Les gains de tous les joueurs sont associés aux terminaisons ou feuilles de l'arbre.

Donnons un exemple de cadre de jeu :

Soit un jeu X, ayant deux *joueurs.* Ces derniers peuvent réaliser deux *coups* : U ou D pour gagner. Voici les différents gains :

- Si les deux joueurs choisissent U : aucun point
- Si un joueur choisit U et l'autre choisit D : 2 points pour le joueur U et 1 point pour le joueur 1.
- Si les deux joueurs choisissent D : le premier D gagne 3 points et le deuxième 1 point.

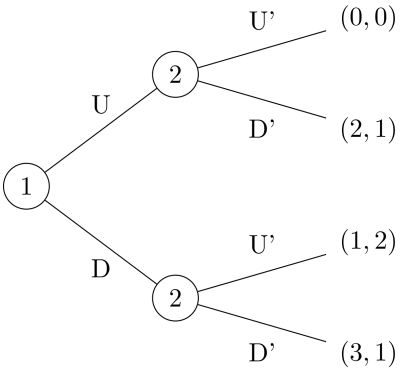


Schéma de la forme extensive du jeu X

Limite de la forme extensive : A l'instar de la forme stratégique, ici il y a une dimension de temps. En effet les joueurs agissent après coup et de manière successive. Donc chaque joueur adapte sa stratégie future et présente en fonction de l'autre joueur.

Par exemple, le joueur 2 a intérêt de jouer :

- le D si le joueur 1 a choisi U
- U si le joueur 2 a choisi D

On peut remarquer que on a plus de libertés avec la forme extensive que la forme normale. Il peut être utile de savoir que un jeu sous forme normale peut avoir plusieurs forme extensive mais un jeu sous forme extensive peut ne pas avoir de forme normale.

3. Exemple 1 : Le dilemme du prisonnier

Bonnie et Clyde, deux criminels notoires, sont suspectés par la police d'avoir commis un vol à main armée. Ils sont arrêtés et interrogés séparément. Manquant de preuves, les policiers cherchent à les faire avouer. Pour cela, ils proposent à chacun l'arrangement suivant : le premier dénonce son complice sans être dénoncé à son tour, il est relâché et son compère écope de 10 ans de prison. Si les deux individus coopèrent (dénoncent), chacun sera condamné à 5 ans de prison ; mais si les deux se taisent, ils purgent une peine de six mois seulement.

Le cadre de ce jeu permet une forme normale par conséquent aussi une forme extensive.

Forme extensive:

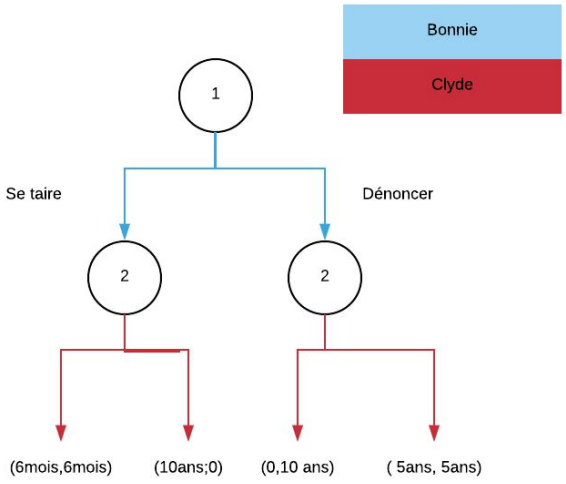


Schéma de la forme extensive du dilemme du prisonnier

Forme Normale/Stratégique :

	Bonnie		
Clyde		Se taire	Dénoncer
	Se taire	(6 mois , 6 mois)	(0 , 10 ans)
	Dénoncer	(10 ans,0)	(5 ans, 5 ans)

Matrice : Forme Normale /Stratégique du Dilemme du prisonnier

Il est important de préciser que les coups de Bonnie et Clyde se font simultanément et pas successivement. La forme normale est donc plus adapté pour voir la simultanéité.

4. Exemple 2 : Jeu de la guerre des sexes

Un couple s'est donné rendez-vous pour la soirée, mais aucun ne parvient à se souvenir si c'est pour assister à un match de foot ou aller à l'opéra. Le mari préférerait aller voir le foot, la femme aimerait aller à l'opéra. Tous deux préfèrent cependant aller au même endroit plutôt que d'être seuls. S'ils ne peuvent pas communiquer, où iront-ils ?

Si on formalise en attribuant des gains à chaque issue, on peut supposer :

- +1 si l'homme va au match
- +1 si la femme va à l'opéra
- +1 si les deux sont au même endroit

Forme normale / stratégique :

	Homme		
Femme		Match	Opéra
	Match	(2,1)	(0,0)
	Opéra	(1,1)	(1,2)

Matrice de la forme stratégique du jeu de la guerre des sexes

2. Cellules et Cellules tumorales

1. Comparaison classique du comportement d'une cellule tumorale et une cellule saine

a. Prolifération

Les cellules saines prolifèrent en cas de nécessité. La prolifération chez les cellules saines ne s'enclenchent que lorsqu'elle reçoivent des signaux extérieurs de proliférations afin de respecter l'équilibre tissulaire. On peut donner comme exemple de facteur de croissance l'EGF : Epithélial Growth Factor ou Facteur de croissance épidermique. L'EGF[1] est une protéine créer par des tissus,

[1] Voir Annexe 1 : pour plus de détails sur le fonctionnement de l'EGF

en général des tissus rénaux. Une cellule saine respecte alors un cycle cellulaire [2]composé de 4 phases : G1, S, G2 et M.

Les cellules cancéreuses court-circuitent le prolifération cellulaire en produisant eux même leurs propre facteurs de croissance. Les cellules cancéreuse vont sur- activer leurs gènes de division cellulaire : oncogènes. Ces derniers vont permettre de produire des protéines qui agissent comme des facteurs de croissances et qui se libèrent dans le milieu extracellulaire. Libérés dans le milieu, ils peuvent activer la division cellulaire des cellules voisine et sa propre division.

Voici un tableau récapitulatif de la prolifération cellulaire :

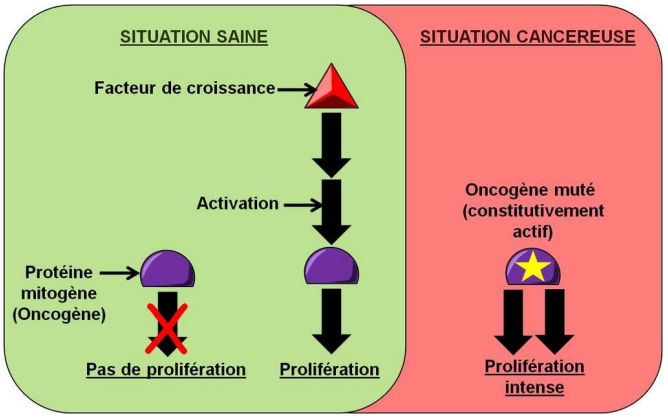


Schéma de la prolifération cellulaire

b. Durée de vie

Une cellule saine a une durée de vie définie dès sa naissance. Toute les cellules sont programmées pour mourir après un certain nombre de divisions.
A chaque division cellulaire, le télomère se raccourcit. L'apoptose se déclenche lorsqu'il n'y a plus de télomère[3].

Une cellule cancéreuse a la particularité d'avoir la télomérase, une enzyme qui permet de conserver la longueur du chromosome. Par conséquent une cellule tumorale peut se diviser de manière indéfini et ne jamais atteindre son apoptose.

[2] Voir Annexe 2 pour plus de détails sur le cycle cellulaire d'une cellule saine
[3] Télomère : gènes aux extrémité des chromosomes

Voici un schéma récapitulatif de la durée de vie d'une cellule saine contre tumorale.

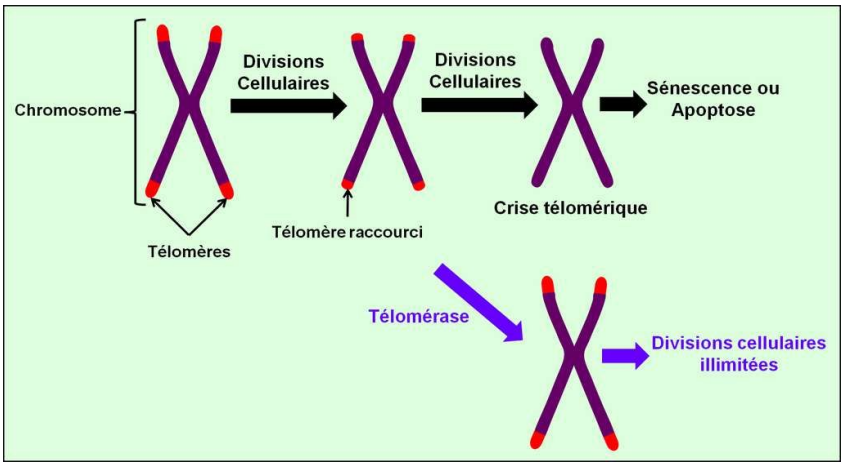


Schéma de la durée de vie des cellules

c. Comportement avec les cellules voisines

Il existe un mécanisme d'homéostasie [4] entre les cellules saines. Une cellule mère se divise en deux cellules filles si la cellules voisine meurt. Cela permet un volume de tissu constant et un apport en nutriment homogène au niveau des cellules.

Les cellules tumorales ne respectent pas ces règles et se divisent indépendamment du tissu et des cellules voisines. Il est donc pas anodin d'observer des amas de cellules tumorales. Cependant certaines peuvent mourir par étouffement et privation de nutriment.

d. Mobilité

Contrairement à une cellule saine fixée sur son tissu, la cellule cancéreuse est mobile. Elle émet des pseudopodes[5] lui permettant de se déplacer.

e. Effraction des tissus

Cette capacité, très complexe techniquement permet à la cellule tumorale de pénétrer dans les tissus. Cette propriété lui permet notamment de s'introduire dans la lumière des capillaires sanguins et de se laisser emporter par le courant sanguin, c'est la 1ère étape du processus métastatique et par conséquent de se propager dans d'autres tissus.

[4] Homéostasie : Auto-régulation
[5] Pseudopode : Déformations de la membrane plasmique qui permettent à une cellule de se nourrir et se déplacer en rampant sur un support dans une direction déterminée

2. Comparaison du comportement d'une cellule tumorale face à une autre cellule tumorale

Il est important de noter que les caractéristiques tumorales ne surviennent pas en même temps. Ces caractéristiques sont liés à des mutations génétiques qui sont due au hasard et l'exposition de facteurs cancérigènes. Une cellule tumorale adopte ces caractéristiques au bout de plusieurs mutations.Une tumeur a besoin d'exhiber plusieur qualités avant d'arriver à un état malin.

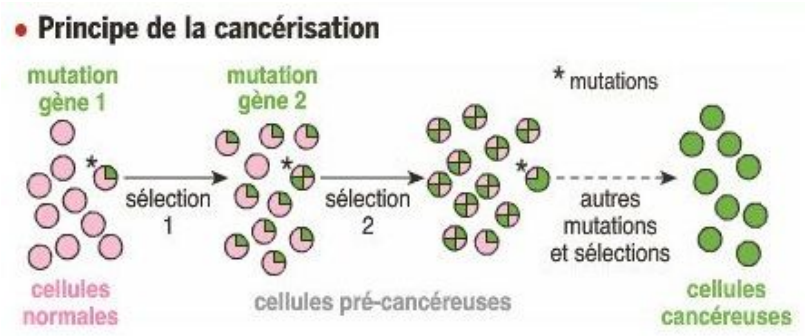


Schéma du principe de la cancérisation

L'article *Game Theoretical Model of Cancer Dynamics with four cell phenotypes* admet qu'il existe une variété de cellules tumorales et plusieurs phénotypes. C'est d'ailleurs à cause de cette diversité en phénotype qu'il est difficile de les traiter et éliminer.

Basé sur le modèle de la sélection naturelle, contrairement à ce qu'on pensait avant qu'il y aurait une compétition entre les cellules tumorales et les cellules saines, la compétition s'installe entre les différents phénotypes des cellules tumorales pour leur survie.

Le rapport de Hanahan et Weinberg détermine 10 différents phénotypes de base qui interviennent dans la propagation de la tumeur et qui sont en compétition.

On a :

- en vert: les cellules qui produisent des facteurs de proliférations
- en gris : les cellules qui résistent à la mort cellulaire

- en rouge : les cellules qui permet l'angiogenèse [6]
- en bleu clair : les cellules immortelles et qui se répliquent en cellules immortelles
- en noir : les cellules qui activent la métastase
- en marron : les cellules qui a la possibilité d'éviter des facteurs qui peuvent la contrôler
- en bleu foncé : les cellules résistantes au principe actif PARP
- en jaune : les cellules qui résistent au réaction anti inflammatoire
- en violet : les cellules qui dérèglent l'apport énergétique des cellules
- en rose les cellules qui évitent la destruction par le système immunitaire

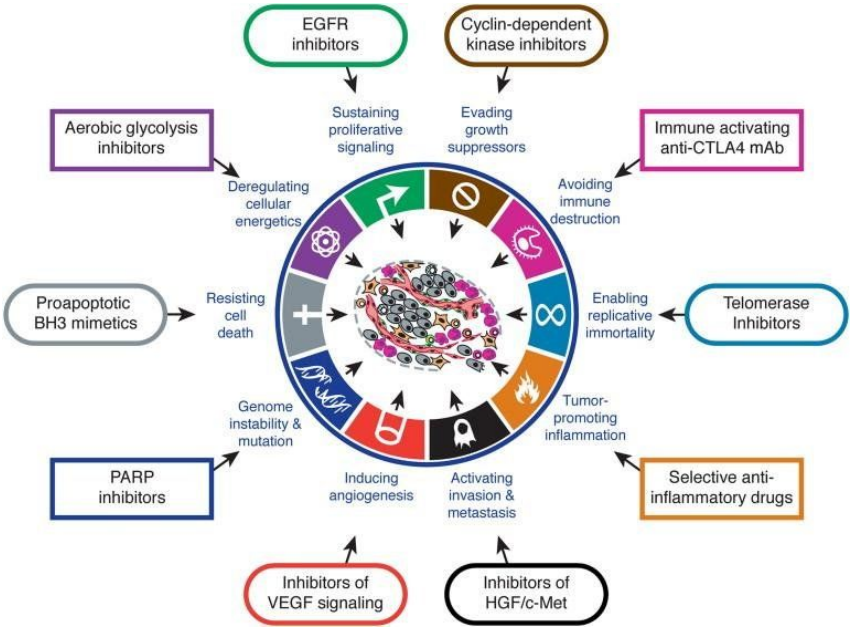


Schéma du modèle de Hanahan et Weinberg

Toutes ces cellules sont en compétition et développent des stratégies pour gagner en espace, nutriments et donc pour leur survie. Ces cellules tumorales forment un amas hétérogène qui peuvent soit coopérer ensemble soit s'autodétruire.

Voici un schéma bilan du amas type composé de cellules tumorales:

[6] Angiogenèse : processus de croissance de nouveaux vaisseaux sanguins à partir de vaisseaux préexistants et permet l'apport en nutriments.

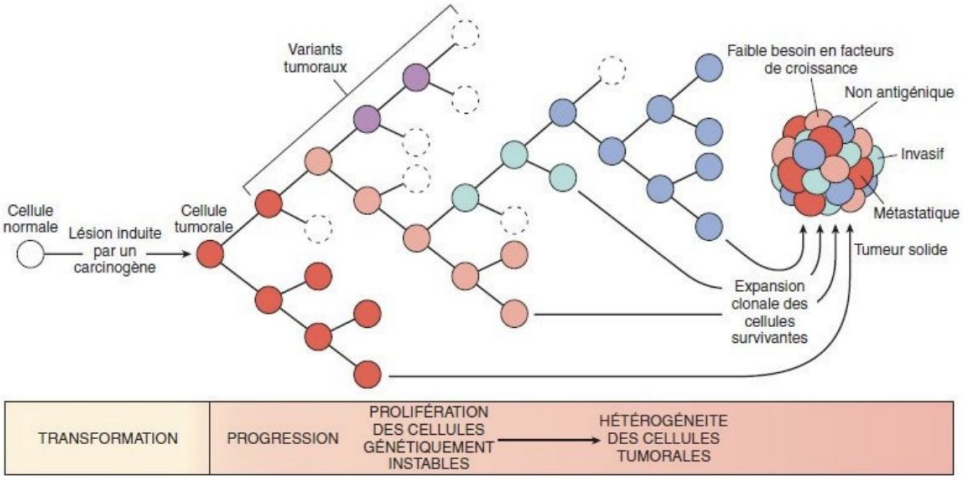


Schéma de formation d'un amas cellulaire

3. Modèle des théories des jeux appliquée aux cellules tumorales

L'article *Game Theoretical Model of Cancer Dynamics with four cell phenotypes* de Elena Hurlbut, Ethan Ortega, Igor V. Erovenko et Jonathan T. Rowell a proposé un algorithme basé le rapport de Hanahan et Weinberg.
Son algorithme contrairement au modèle de Hanahan Weinberg basé sur 10 différents phénotypes cellulaires tumoraux, se base sur 4 phénotypes:

- A+ : Cellules productrices de facteurs d'anxiogène qui permet la vascularisation de la tumeur
- C : Cellules cytotoxiques : elles tuent les autres cellules et sont immunisées face aux productions des autres cellules cytotoxiques
- A- : Cellules stromales neutres
- P : Cellules prolifératives

Ils étudient par la suite 6 interactions possibles ce qui correspond à 6 théories de jeux différents entre :A+/A+ ; A+/C ; A+/A- ; A+/P ; C/C ; C/A- ; C/P ; A-/A- ; A-/P ; P/P

Cependant, pour des buts de simplification, j'ai décidé de me concentrer sur 2 phénotypes : Les cellules prolifératives P et les cellules stromales neutres A-. Mais on peut appliquer la même stratégie pour les autres phénotypes.

Ils ont attaché pour ces interactions différents paramètres[7]. L'interaction A-/P se résume à un seul paramètre : g : les avantages reproductifs d'une cellule proliférative.

On peut alors poser le jeu : Soit X1,X2 : deux cellules voisines. Ces cellules sont déterminées par leur phénotype qui résume leur coup et leur stratégie. Soit X peut être une cellule stromale A-, soit une cellule proliférative P.

Voici les règles :

- cellule stromale A1- ne gagne rien d'être voisine avec une A2- ou une cellule P2.
- cellule proliférative P1 gagne toujours en capacité de prolifération avec une cellule P2 ou A2-.

On peut alors poser la matrice des bénéfices donc la forme Normale / Stratégique de jeu tumorale.

	Opposants		
Joueurs		A2-	P2
	A1-	1	1
	P1	1+g	1+g

Matrice des bénéfices A-/P

On peut poser les différents équilibres de Nash :

Sortie	Condition
A- domine	g<0
P domine	g>0
Exclusion	N/A
Cohabitation	g=0

Tableau des différents équilibres de Nash

[7] Voir annexe 4

4. Algorithmes

1. Le Jeu de la vie de Conway

Pour modéliser la propagation en python des cellules tumorales on a choisi le fameux Jeu de la vie. Le Jeu de la vie est un automate cellulaire imaginé par John Horton Conway.

J'ai utilisé un algorithme[8] déjà présent sur internet en python comme la modélisation de ce jeu n'étant pas dans le cadre de ma TX.

Le jeu de la vie se passe sur une grille 2D avec longueur et largeur définie et donc un nombre de cases définies. Une case correspond à une cellule vivante. Cette cellule peut avoir alors deux états : soit vivante (état 1) soit morte (état 0).

Le principe est assez simple :

- La valeur d'une case passe à 0 si elle possède 0 ou 4 voisins à 1 (elle meurt isolée ou étouffée)
- La valeur d'une case passe 1 si elle possède 2 ou 3 voisins à 1 (elle renaît)
- Elle ne change pas dans les autres cas.

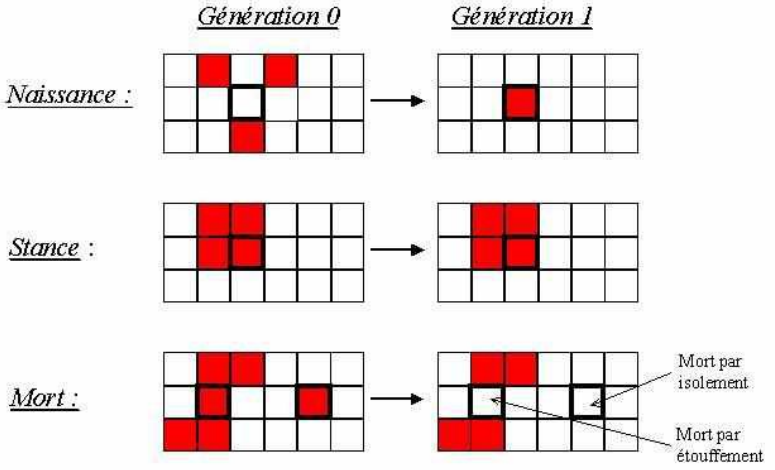


Schéma du principe du Jeu de la vie

Explication du programme :

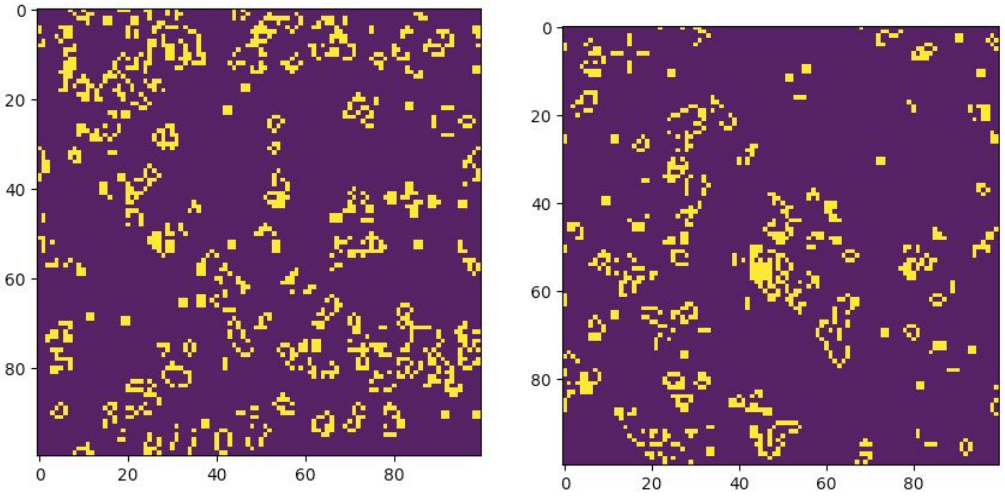
[8] Programme en Python en Annexe 3.

- Si une cellule est vivante on attribue la valeur 255 à la case
- Si une cellule est morte on attribue la valeur 0 à la case
- def randomGrid (N) : Cette fonction retourne une matrice de taille N X N définie en paramètre. A l'intérieur de cette matrice on a donc N^2 cases. La fonction attribue de manière aléatoire 0 ou 255 à chacune des cases avec une probabilité attachée à chacune des valeurs définie dans le programme .
- def update(framenum, img, grid, N): A chaque update on produit une nouvelle grille avec les nouvelles valeurs dans chaque case. On réalise 2 boucle for imbriquées permettant de scanner toutes les cellules. La première boucle for scanne toutes les lignes de la grille et le deuxième boucle à l'intérieur scanne pour chaque toutes les colonnes donc toutes les cases.

  A l'intérieur de ces deux boucles, on somme les valeurs des 8 cases aux alentours. On exécute par la suite les règles de conway.

  Si pour une cellule vivante avec pour coordonnées [i,j] a un total inférieur à 2 ou supérieur à 3, la cellule meurt. Sinon si pour une cellule morte avec pour coordonnées [i,j] a un nombre de voisines égale à 3, la cellule renaît et on l'attribue la valeur ON.

Quelques exemples de simulations :



Exécution du programme

2. Algorithme de propagation des tumeurs

On pose trois phénotypes choisis : les cellules stromales normales A- ,les cellules prolifératives P et les cellules cytotoxiques C. Nos cellules vivent dans une matrice bidimensionnelle 100 x 100. Chaque cellule interagit simultanément avec ces 8 cellules voisines. A l'initialisation, le tissu est sain et est donc composé que de cellules A-. 0,2% des cellules sont choisies au hasard et on les attribue le phénotype P et le phénotype C. Chaque cellule est caractérisée par sa capacité à se reproduire et à mourire

On peut poser que :

- Les cellules stromales ont une capacité de reproduction de 1
- La capacité de mourir de toutes les cellules est de 1
- Les cellules P ont une capacité de reproduction de 1+g[9]
- Les cellules C ont une capacité de reproduction de 1-b
- A chaque interaction avec une cellule cytotoxique, la capacité de mourire augmente de c pour une cellule stromale et de c(1+g) pour les cellules prolifératives

L'algorithme se base sur ces causes conséquences :

- La cellules avec une forte capacité de division prendra la place de la cellule voisine avec une forte capacité de mortalité
- La cellule avec la plus forte capacité de mortalité est remplacée par la cellule avec la plus grande capacité de division

On peut résumer sous forme de tableau :

Phénotypes	Capacité de se diviser	Capacité de mourire
A-	1	1+c
P	1+g	(1+c)(1+g)
C	1-b	1

Tableau résumant les capacités de division et de mourire

[9] Ce sont les paramètres vus p.13, la liste exhaustive est en Annexe 4

3. [10]Programme en Python

L'algorithme se déroule ainsi :

- def RandomGrid(N) : La fonction produit la matrice initiale et attribue une probabilité modifiable à chaque phénotype. J'ai attribué une probabilité de 0.996 pour les cellules saines, 0.002 pour les cellules cytotoxiques, 0.002 pour les cellules prolifératives et 0 pour les cellules mortes.
- def Update: Cette fonction a pour but de mettre à jour la matrice en créant une nouvelle matrice. Dans cette fonction, je définis alors les règles d'interactions :
  - Si la cellule est morte :
    - Si les cellules voisines sont prolifératives : la cellule proliférative se divise et la cellule morte devient proliférative
  - Si la cellule est soit saine ou proliférative :
    - Si les cellules voisines sont du type cytotoxiques : la cellule meurt
- Le principe reste le même cependant j'ai du enlevé les paramètres énoncés car il aurait fallu créer une matrice de 100 x 100 cellules. Chaque cellule devait être défini par un vecteur tridimensionnel. L'algorithme aurait été bien plus compliqué.

4. Résultats

Voici une évolution typique des cellules tumorales.

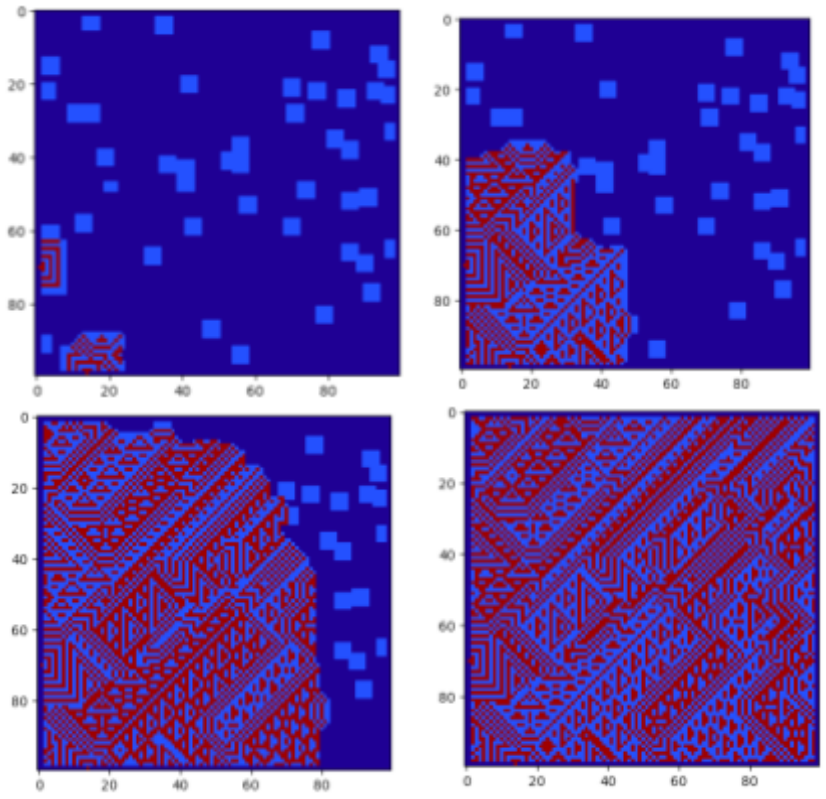
Pour des raisons esthétiques et de clarté, les phénotypes tumoraux sont tous rouge. En bleu foncé ce sont les cellules saines et en bleu clairs les cellules mortes.

On peut observer plusieurs choses :

- Il est important de garder en mémoire que le tissu n'avait à l'origine aucune cellules mortes donc les cubes bleus clairs présents à la première image sont des cellules tuées par des cellules cytotoxiques.

[10] Voir le programme en Annexe 5

- On observe aussi que malgré la présence de plusieurs foyers locaux de cellules tumorales, c'est en général 2 ou 3 foyers de cellules tumorales locales au maximum qui se propagent plus largement.



Exécution du programme

5. Conclusion

Durant ce travail de laboratoire, on a pu étudié la théorie des jeux, le mode de propagation des cellules tumorales séparément et regrouper ces deux théories sous la forme d'un programme en Python. Ce travail très enrichissant m'a permis d'en apprendre plus de manière autodidacte. Mes connaissances en cellules tumorales ou en Python étant très basiques, j'ai pu apprendre à par exemple créer des animations en Python, trier les articles scientifiques et entreprendre un projet proche de mon domaine.

Ce travail de laboratoire peut se prolonger. En effet, il est possible comme je l'ai énoncé plus haut d'intégrer les paramètres issus de chaque phénotypes afin d'avoir peut-être un résultat plus réaliste.

Il serait aussi intéressant d'intégrer d'autres phénotypes tumoraux ou de passer d'une matrice en 2D en un modèle en 3D afin de voir la propagation des cellules tumorales sur plusieurs couches.

6. Bibliographie

1. Cours de Sébastien Konieczny de l'Université d'Artois-Lens : Introduction à la théorie des jeux: http://www.cril.univ-artois.fr/~konieczny/enseignement/TheorieDesJeux.pdf
2. Généralités sur les tumeurs - Collège français des Pathologistes (Copath) http://campus.cerimes.fr/anatomie-pathologique/enseignement/anapath_7/site/html/cours.pdf

3. Cycle cellulaire : prolifération des cellules saines - FuturaSanté https://www.futura-sciences.com/sante/dossiers/medecine-cancer-mecanismes-biologiques-1453/page/6/

4. Librairie des molécules EGF : http://www.librairiedemolecules.education.fr/molecule.php?idmol=214

5. Cycle Cellulaire : prolifération des cellules cancéreuses https://www.futura-sciences.com/sante/dossiers/medecine-cancer-mecanismes-biologiques-1453/page/7/

6. Article de l'EPFL : Cancer : des chercheurs décryptent l'immortalité cellulaire https://actu.epfl.ch/news/cancer-des-chercheurs-decryptent-l-immortalite-c-2/

7. Game Theoretical Model of Cancer Dynamics with Four Cells phenotypes de Elena Hurlbut, Ethan Ortega, Igor V. Erovenko, Jonathan T. Rowell https://www.mdpi.com/2073-4336/9/3/61

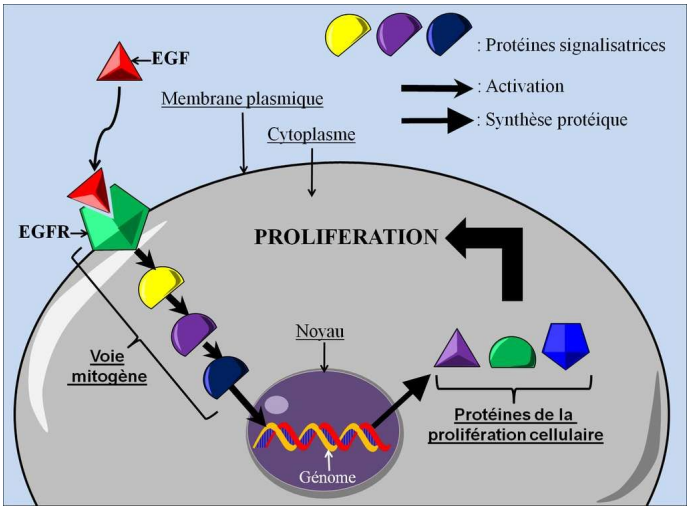
8. Geeks for geeks : Algorithme de Conway https://www.geeksforgeeks.org/conways-game-life-python-implementation/

9. Hallmarks of cancer : The Next Generation de Douglas Hanahan et Robert A. Weinberg
https://www.cell.com/fulltext/S0092-8674(11)00127-9

7. Annexe

Annexe 1 : Fonctionnement de l'EGF

L'EGF va se fixer sur le récepteur EGFR de la cellule. Cette voie déclenche en cascade de nombreuse réactions appelée la voie mitogène. Ces signaux arrivent au niveaux du noyau de la cellule et va activer des gènes de la mitose. Par la suite l'ADN va produire via l'ARN des protéines de la prolifération cellulaire et va permettre la division de la cellule.



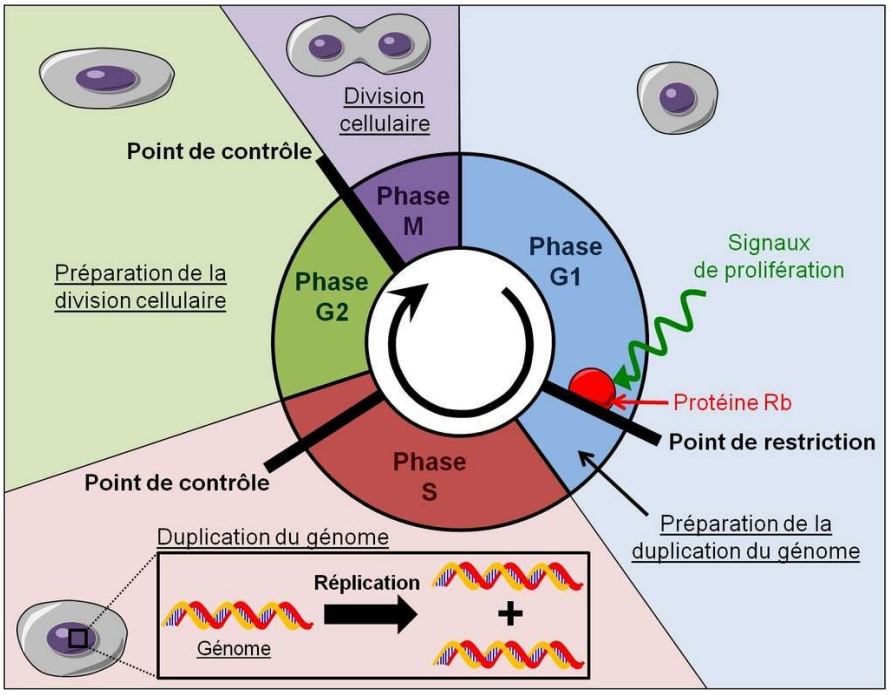
https://www.futura-sciences.com/sante/dossiers/medecine-cancer-mecanismes-biologiques-1453/page/6/

Annexe 2 : Cycle Cellulaire d'une cellule saine

La phase G1 dure quelques heures à plusieurs années, elle consiste à préparer la cellule à la réplication.
La phase S dure entre 6 à 20h et permet la réplication de l'ADN.
La phase G2 dure entre 2 et 6h correspond à la préparation à la mitose.

La phase M dure 1 à 2h correspond à la mitose.

Il existe aussi une phase G0 avant la phase G1 qui correspond à la phase hors du cycle de division cellulaire.



https://www.futura-sciences.com/sante/dossiers/medecine-cancer-mecanismes-biologiques-1453/page/6/

Annexe 3 : Programme de Conway

```
# Python code to implement Conway's Game Of Life
# Python code to implement Conway's Game Of Life
import argparse
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation

# setting up the values for the grid
ON = 255
OFF = 0
vals = [ON, OFF]


def randomGrid(N):
    """returns a grid of NxN random values"""
```

```
    return np.random.choice(vals, N * N, p=[0.2, 0.8]).reshape(N, N)



def update(frameNum, img, grid, N):
    # copy grid since we require 8 neighbors
    # for calculation and we go line by line
    newGrid = grid.copy()
    for i in range(N):
        for j in range(N):

            # compute 8-neghbor sum
            # using toroidal boundary conditions - x and y wrap around
            # so that the simulaton takes place on a toroidal surface.
            total = int((grid[i, (j - 1) % N] + grid[i, (j + 1) % N] +
                         grid[(i - 1) % N, j] + grid[(i + 1) % N, j] +
                         grid[(i - 1) % N, (j - 1) % N] + grid[(i - 1) % N, (j + 1) % N] +
                         grid[(i + 1) % N, (j - 1) % N] + grid[(i + 1) % N, (j + 1) % N]) / 255)

            # apply Conway's rules
            if grid[i, j] == ON:
                if (total < 2) or (total > 3):
                    newGrid[i, j] = OFF
            else:
                if total == 3:
                    newGrid[i, j] = ON

                # update data
    img.set_data(newGrid)
    grid[:] = newGrid[:]
    return img,


# main() function
def main():
    # Command line args are in sys.argv[1], sys.argv[2] ..
    # sys.argv[0] is the script name itself and can be ignored
    # parse arguments
        parser = argparse.ArgumentParser(description="Runs Conway's Game of
Life simulation.")

    # add arguments
    parser.add_argument('--grid-size', dest='N', required=False)
```

```
    parser.add_argument('--mov-file', dest='movfile', required=False)
    parser.add_argument('--interval', dest='interval', required=False)
    parser.add_argument('--gosper', action='store_true', required=False)
    args = parser.parse_args()

    # set grid size
    N = 100
    if args.N and int(args.N) > 8:
        N = int(args.N)

    # set animation update interval
    updateInterval = 50
    if args.interval:
        updateInterval = int(args.interval)

    # declare grid
    grid = np.array([])

    # check if "glider" demo flag is specified

    grid = randomGrid(N)

    # set up animation
    fig, ax = plt.subplots()
    img = ax.imshow(grid, interpolation='nearest')
    ani = animation.FuncAnimation(fig, update, fargs=(img, grid, N,),
                                  frames=10,
                                  interval=updateInterval,
                                  save_count=50)

    # # of frames?
    # set output file
    if args.movfile:
        ani.save(args.movfile, fps=30, extra_args=['-vcodec', 'libx264'])

    plt.show()


# call main
if __name__ == '__main__':
    main()
```

Annexe 4 : Table des paramètres

a	Coût de production de facteurs d'angiogenèse
b	Coût de production de cytotoxines
c	Coût d'interaction avec des cytotoxines
d	Bénéfices quand on interagit avec des A+
e	Bénéfices pour les cellules C à détruire les autres cellules
f	Bénéfices interactions entre A+/A+
g	Avantage reproductif des cellules P

Annexe 5 : Programme finale en Python

```
import argparse
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import matplotlib.cm as cm
SAINE=0
CYTO=100
PROLI=100

MORT=20
#BLEU = VIVANT, ROUGE = PROLI et CYTO ,BLEU CLAIR = MORT
vals = [SAINE, CYTO,PROLI, MORT]

def randomGrid(N):
  """returns a grid of NxN random values"""
  return np.random.choice(vals, N * N, p=[0.996, 0.002,0.002,0]).reshape(N, N)
def update(frameNum, img, grid, N):

  newGrid = grid.copy()
  for i in range(2,N-1):
    for j in range(2,N-1):

        newGrid[i,j]= grid[i,j]
        if (grid[i,j]==MORT):
```

```
            if ((grid[i-1,j]==PROLI) or (grid[i+1,j]==PROLI) or (grid[i,j-1]==PROLI) or
(grid[i,j+1]==PROLI) or (grid[i-1,j-1]==PROLI)
                                or  (grid[i-1,j+1]==PROLI)  or  (grid[i+1,j+1]==PROLI)  or
(grid[i+1,j-1]==PROLI)):
                newGrid[i,j]=PROLI

        if ((grid[i,j]== SAINE) or (grid[i,j]==PROLI)):
            if ((grid[i - 1, j]==CYTO) or (grid[i+1, j]==CYTO) or (grid[i, j-1]==CYTO
)or (grid[i, j+1]==CYTO )or (grid[i-1, j-1]==CYTO)
            or (grid[i - 1, j + 1] ==CYTO) or (grid[i + 1, j + 1] ==CYTO) or (grid[i + 1,
j - 1] ==CYTO)) :
                newGrid[i,j]=MORT

            # update data
    img.set_data(newGrid)
    grid[:] = newGrid[:]
    return img,

def main():
    # Command line args are in sys.argv[1], sys.argv[2] ..
    # sys.argv[0] is the script name itself and can be ignored
    # parse arguments
      parser = argparse.ArgumentParser(description="Runs Conway's Game of
Life simulation.")

    # add arguments
    parser.add_argument('--grid-size', dest='N', required=False)
    parser.add_argument('--mov-file', dest='movfile', required=False)
    parser.add_argument('--interval', dest='interval', required=False)
    parser.add_argument('--gosper', action='store_true', required=False)
    args = parser.parse_args()

    # set grid size
    N = 100
    if args.N and int(args.N) > 8:
        N = int(args.N)

    # set animation update interval
    updateInterval = 50
    if args.interval:
        updateInterval = int(args.interval)


    grid = np.array([])
```

```
    grid = randomGrid(N)

    # set up animation
    fig, ax = plt.subplots()
    assert isinstance(ax.imshow, object)
    img = ax.imshow(grid,cmap=cm.jet, interpolation='nearest') #Montre l'image
    ani = animation.FuncAnimation(fig, update, fargs=(img, grid, N,),
                                  frames=10,
                                  interval=updateInterval,
                                  save_count=50)

    # # of frames?
    # set output file
    if args.movfile:
        ani.save(args.movfile, fps=30, extra_args=['-vcodec', 'libx264'])

    plt.show()


# call main
if __name__ == '__main__':
    main()
```