CFPT-INFORMATIQUE

Travaux de Diplômes 2017

Dilemme du Prisonnier

Automate cellulaire

Julien SEEMULLER

Supervisé par :

Mme. Terrier

T.IS-E2B

11 avril 2017

1 Abstract

The prisoner's dilemma is a well known paradox that demonstrates why two completely rational individuals might act in their own self-interest even if their course of action does not result in the ideal outcome. [1]

A cellular automaton is a set of colored cells placed on a grid that evolves according to rules based on the state of each cell's neighbors. Cellular automata are often used to model phenomena present in the physical world. Depending on the their level of accuracy, some can even be used for making predictions in the physical world. [2]

During my final year's project work, I will attempt to modelize the *iterated* prisoner's dilemma (a repeated version of the prisoner's dilemma) using a cellular automaton in C#. The results of my work will try to establish the most reliable strategy to use while "playing" the *iterated* prisoner's dilemma.

2 Résumé

Le dilemme du prisonnier est un paradoxe connu démontrant pourquoi deux individus considérés comme "rationnels" peuvent agir dans leur propre intérêt, même si leurs actions ne leur apportent pas forcement un résultat idéal. [1]

Un automate cellulaire est un ensemble de cases colorées se trouvant dans une grille. Ces cellules évoluent en se basant sur des règles définies par l'état des voisins de chaque cellule. Les automates cellulaires sont couramment utilisés comme moyen de modéliser des situations présentes dans le monde réel. Suivant le niveau de précision de ces derniers, il est même envisageable de tirer des conclusions de leurs résultats. [2]

Lors de mon travail de diplôme, j'ai pour but de modéliser le dilemme du prisonnier *répété* en C# à l'aide d'un automate cellulaire. Les résultats de mon travail permettront d'établir la stratégie la plus optimale en "jouant" au dilemme du prisonnier *répété*.

Table des matières

3 Cahier des charges

3.1 Sujet

Automate cellulaire (voir *Conway's Game of Life* [3]) basé sur le dilemme du prisonnier répété (voir *Iterated Prisoner's Dilemma* [4]) et permettant de le simuler.

3.2 Descriptions

Le projet étant basé sur des concepts peu courants, il est nécessaire de les détailler.

Automate cellulaire :

Un automate cellulaire est un modèle constitué d'une grille de cellule changeant d'état à chaque temps $t+1$. Une règle est appliquée à toutes les cellules, habituellement basée sur l'état des voisins de chaque cellule, et permet de faire "évoluer" la grille. L'automate cellulaire le plus connu est probablement *Game of Life* imaginé par John Horton Conway en 1970.

Dilemme du prisonnier répété :

Le dilemme du prisonnier répété est une variante du dilemme du prisonnier. Dans ce jeu, des personnes jouent plusieurs fois au dilemme du prisonnier.

Dans le dilemme du prisonnier, deux prisonniers ayant commis un crime mineur sont enfermés dans deux cellules différentes, afin de les empêcher de communiquer. Le policier soupçonne les deux accusés d'avoir commis auparavant un crime plus important et souhaite obtenir des aveux concernant ce dernier. Il se présente donc et discute avec chaque prisonnier séparément en leur offrant à chacun deux choix :

- Dénoncer l'autre prisonnier (trahison)
- Se taire (coopération)

Il présente donc les résultats des choix suivants :

- Si l'un des deux prisonniers dénonce l'autre, il est remis en liberté alors que le second obtient la peine maximale (10 ans)
- Si les deux se dénoncent entre eux, ils seront condamnés à une peine plus légère (5 ans)
- Si les deux refusent de dénoncer l'autre, la peine sera minimale (6 mois), faute d'éléments à charge.

Chaque prisonnier fait donc une *"Matrice des Gains"* [5] pour résoudre ce problème :

	Il se tait	**Il me dénonce**
Je me tais	(-1/2 ; -1/2)	(-10 ; 0)
Je le dénonce	(0 ; -10)	(-5 ; -5)

Chaque prisonnier *devrait* donc comprendre que le choix logique sur une seule itération est de coopérer avec l'autre prisonnier.

3.3 But

Le but du projet est donc de fusionner ces deux concepts et de créer un automate cellulaire permettant de visualiser le dilemme du prisonnier répété. Chaque cellule jouerait une partie du dilemme simultanément avec chacun de ses voisins. Chaque cellule peut adopter une stratégie permettant d'optimiser ses gains. Voici quelques exemples de stratégies :

Random (RAND) : Fait des actions aléatoires, trahit ou coopère avec 50% de chance.
Always Defect (AD) : Trahit avec 100% de chance.
Always Cooperate (AC) : Coopère avec 100% de chance
Grim Trigger (GRIM) : Stratégie "AC", mais change sa stratégie vers "AD" après trahison.
etcetera... [6]

Beaucoup de stratégies peuvent être implémentées pour rendre le jeux intéressant à étudier. Pour cela, des graphiques seront implémentés permettant de récupérer et d'observer les résultats de l'application en temps réel. Les cellules du plateau seront aussi colorées selon leur stratégie ou encore l'historique de leur actions (ex : tendance à trahir → rouge et tendance à coopérer → vert).

Voici en exemple, le dilemme du prisonnier sous forme d'automate cellulaire :

A	*B*	*A*	*B*	*A*	*B*
B	*A*	*B*	*A*	*B*	*A*
A	*B*	*A*	*B*	*A*	*B*
B	*A*	*B*	*A*	*B*	*A*
A	*B*	*A*	*B*	*A*	*B*
B	*A*	*B*	*A*	*B*	*A*

$T=1$

θ						p					
0	0	0	0	0	0	16	16	16	16	16	16
0	0	0	0	0	0	16	16	**13**	16	16	16
0	0	π	0	0	0	16	**13**	**20**	**13**	16	16
0	0	0	0	0	0	16	16	**13**	16	16	16
0	0	0	0	0	0	16	16	16	16	16	16
0	0	0	0	0	0	16	16	16	16	16	16

$T=2$

θ						p					
0	0	0	0	0	0	16	**13**	16	**13**	16	16
0	π	0	π	0	0	**13**	**20**	**7**	**20**	**13**	16
0	0	π	0	0	0	16	**7**	**20**	**7**	16	16
0	π	0	π	0	0	**13**	**20**	**7**	**20**	**13**	16
0	0	0	0	0	0	16	**13**	16	**13**	16	16
0	0	0	0	0	0	16	16	16	16	16	16

FIGURE 1 – Automate cellulaire du dilemme du prisonnier
source [7]

3.4 Spécifications

Le projet sera conforme aux spécifications suivantes :

Automate cellulaire paramétrable :
- Nombre de cellules paramétrables.
- Stratégies utilisées et proportions de ces dernières sur le plateau paramétrables.
- Matrice des gains [5] paramétrable.

Exploitation des résultats :
- Cellules colorées selon leur stratégie ou historique d'actions.
- Divers graphiques (ex : nombre de cellules traîtres par générations)
- Possibilité d'utilisation de LiveCharts [8]

3.5 Environnement

Le projet prendra place dans l'environnement suivant :
- Ordinateur sous `Windows 7`
- Environnement de développement adapté pour `C#`.

3.6 Livrables

Les documents suivant seront remis à la fin du projet :

- Journal de bord (PDF).
- Rapport technique (PDF).
- Fichier ZIP contenant les sources.

3.7 Reddition

Voici les dates importantes du projet :

22 Janvier 2017 :	Reddition du cahier des charges.
5 Avril 2017 :	Début du travail
à définir :	Rendu du poster.
à définir :	Reddition intermédiaire de la documentation.
12 Juin 2017 :	Reddition finale du projet.
19-20 Juin 2017 :	Présentation orale du projet.

4 Introduction

5 Planification provisionnelle

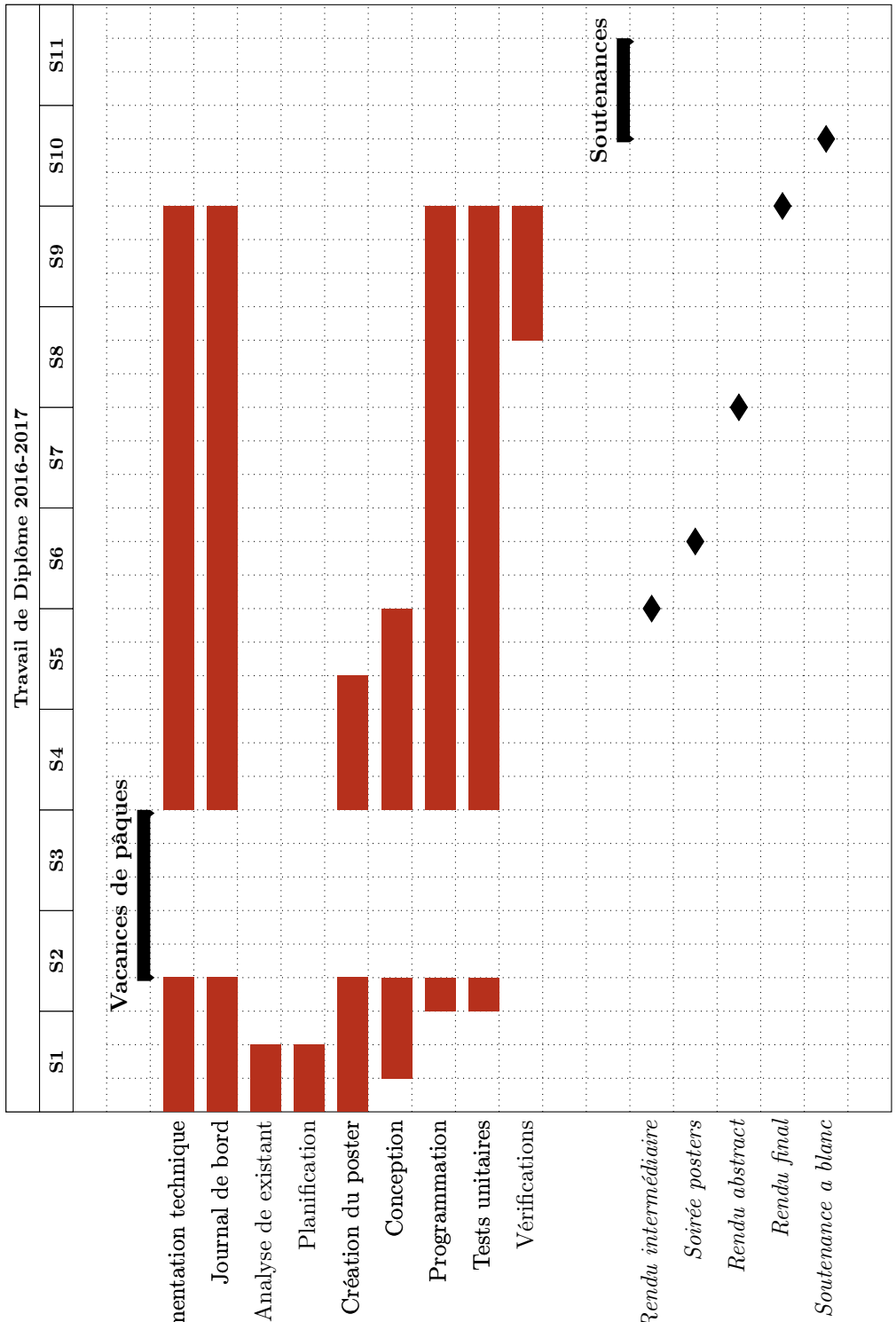

FIGURE 2 – Diagramme de Gantt

6 Analyse de l'existant

Il est nécessaire d'analyser et de comparer différents travaux avant de commencer le développement de notre application. Pour effectuer cette analyse, trois concepts d'automates cellulaires basés sur le dilemme du prisonnier ont étés sélectionnés :

— "*A quantum prisoner's dilemma cellular automaton*" de M. Ramón Alonso-Sanz. [9]
— "*Prisoner's dilemma in one-dimensional cellular automata*" de M. Marcelo Alves Pereira [10]
— "*The prisoner's dilemma and the game of life*" de Mme. Katarzyna Zbieć [11]

6.1 Projet de M. Ramón Alonso-Sanz

Le projet de M. Ramón Alonso-Sanz intitulé "*A quantum prisoner's dilemma cellular automaton*" reprends le dilemme du prisonnier de base, mais y ajoute quelques subtilités :

Le plateau est structuré sous la forme d'un échiquier, chaque cellule a donc quatre alliés et quatre rivaux, comparé à la forme habituelle, qui est d'utiliser les huit voisins de chaque cellules (similaire au mouvements d'un roi dans le jeu d'échecs). Les cellules possèdent des stratégies dites "quantiques" et adaptent aussi leurs stratégies à celle de leurs voisins. Les voisins ayant les meilleures performances sont imités par les autres cellules à l'aide d'une méthode nommé *imitation-of-the best*. Chaque cellule joue aussi avec elle même en plus de ses rivaux. Ceci permet de prendre en compte ses propres résultats en faisant la moyenne des résultats obtenus entre les parties.

Un mécanisme de mémoire est aussi présent dans le programme de M.Ramon Alonso-Sanz. Ce dernier est de type "Markovien" (voir "chaînes de markov"), un historique complet n'est pas stocké mais les résultats et les choix précédents affectent les choix futurs de chaque cellules.

On compare aussi les stratégies dites "quantiques" aux stratégies classiques pour évaluer l'efficacité de ces dernières. Voici a quoi ressemble le projet de M. Ramón Alonso-Sanz :

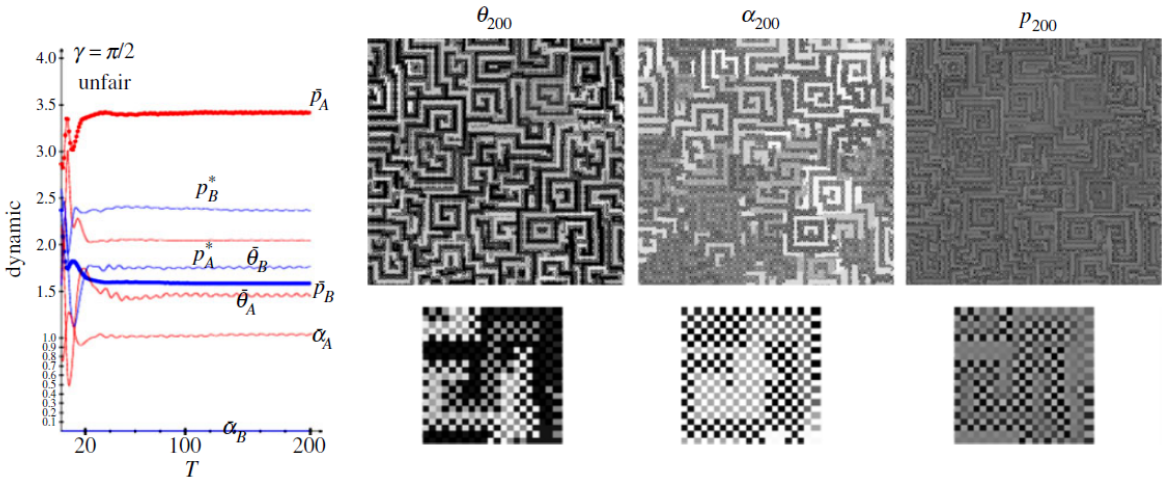

FIGURE 3 – Comparaison de stratégies quantiques (p_A) et classiques (p_B)

6.2 Projet de M. Marcelo Alves Pereira

Le projet de M. Marcelo Alves Pereira possède quelques différences avec un automate cellulaire du dilemme du prisonnier standard. En effet, M. Marcelo Alves Pereira allègue que la majorité des automates cellulaires basés sur le dilemme du prisonnier utilisent des structures trop complexes et suggère ainsi une approche plus simple. Ce dernier modélise le dilemme du prisonnier sous la forme d'un treillis à une dimension (tableau à une dimension), mais sa structure comporte quelques subtilités.

La première subtilité est le fait d'empiler ces tableaux à une dimension pour former un tableau en deux dimensions ou chaque position Y du tableau correspond à un temps T d'une partie. Ce système permet d'avoir en *tout temps* un historique complet et visible de la partie actuelle du dilemme du prisonnier. Voici un schéma représentant le fonctionnement de cette approche :

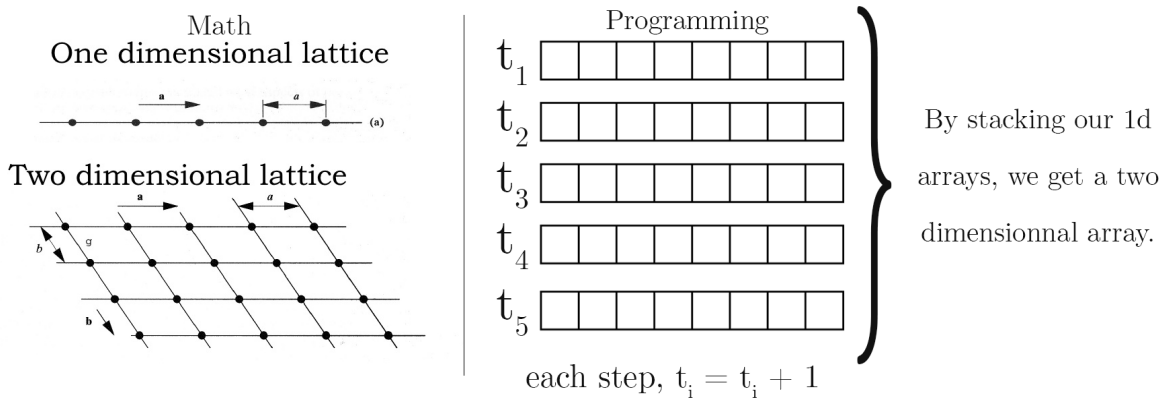

FIGURE 4 – Utiliser la deuxième dimension d'un tableau pour garder un "historique"

La deuxième subtilité est d'utiliser un nombre variable de voisins. En effet, le tableau étant sur un axe unique, on peut représenter le nombre de voisins de chaque cellule simplement par un chiffre X étant pair. Par exemple, pour 6 voisins par cellule, on considère les trois cellules à gauche et a droite de notre cellule actuelle comme nos voisins.

La troisième subtilité est d'utiliser le principe de *self-interaction* (ou "interaction avec soi" en français). Le principe est de "jouer" avec soi-même (la cellule actuelle) pour compenser un manque de joueurs quand le nombre de voisin n'est pas pair (par exemple, sur les bords de la matrice).

Malgré ces subtilités, ce système n'est pas parfait. Les cellules de ce système n'utilisent qu'une seule stratégie : celle du "*tit-for-tat*" (TFT) [6]. Avec cette stratégie, les cellules commencent dans un état aléatoire et copie la stratégie du voisin ayant obtenu le meilleur score. Ainsi, le jeu devient prévisible ; les cellules essaient de maximiser leur gains de manière "égoïste" et des grappes de cellules trahissant leurs voisins se forment rapidement. Ce système n'est pas forcement une mauvaise représentation du dilemme du prisonnier mais il manque cependant de diversité.

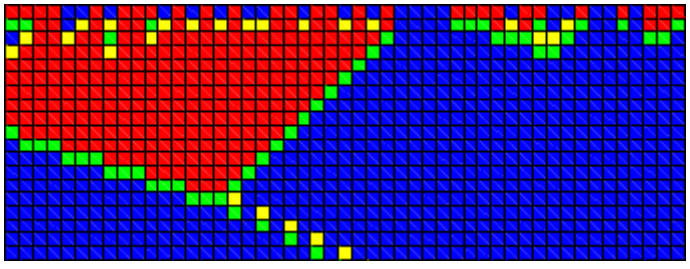

FIGURE 5 – Grappe de cellules "traîtres"

6.3 Projet de Mme. Katarzyna Zbieć

L'approche de Mme. Katarzyna Zbieć est différente des deux projets précédents. Elle vise a combiner le jeu de la vie de Conway et le dilemme du prisonnier. Les différents principes du jeu de la vie (cellules, états, plateau, etc...) et du dilemme du prisonnier (stratégies, matrice de gains, etc...) sont expliqués en détail et par la suite comparés.

Malheureusement, aucun exemple graphique n'est fourni avec le document. Cependant, ce projet reste le plus proche à celui qui sera développé lors de ce travail de diplôme.

Voici un tableau tiré du document de Mme. Katarzyna Zbieć ainsi que sa traduction française. Ces derniers font ressortir les ressemblances entre la structure du jeu de la vie et celle du dilemme du prisonnier :

the Prisoner's Dilemma	the Game of Life
the future of any player depends on the strategy of his/her neighbours	the future of any cell is determined by the state of its neighbours
the players are changing their own strategies in the way determined by the strategies of their enemies	the cells are changing colours in the way determined by the colours of their neighbours
the player can choose one of the two options: to cooperate or to defect	the cell has one of two states: live or dead
strategies	rules

FIGURE 6 – Comparaison entre le jeu de la vie et le dilemme du prisonnier

Le Dilemme du Prisonnier	Le Jeu de la Vie
Le futur de chaque joueur dépends de la stratégie de ses voisins	Le futur de chaque cellule est déterminé par l'état de ses voisins
Les joueurs changent leurs stratégies en se basant sur la stratégie de leurs énemis	Les cellules changent de couleur en accordance avec celles de leurs voisins
Le joueur peut choisir deux options : coopérer ou trahir	La cellule a deux états : vivant ou mort
stratégies	règles

TABLE 1 – Version traduite du tableau des différences entre le *DP* et le *JdlV*

6.4 Conclusions tirées de l'analyse

7 Analyse fonctionnelle

7.1 Fonctionnement

7.2 Diagramme de classe

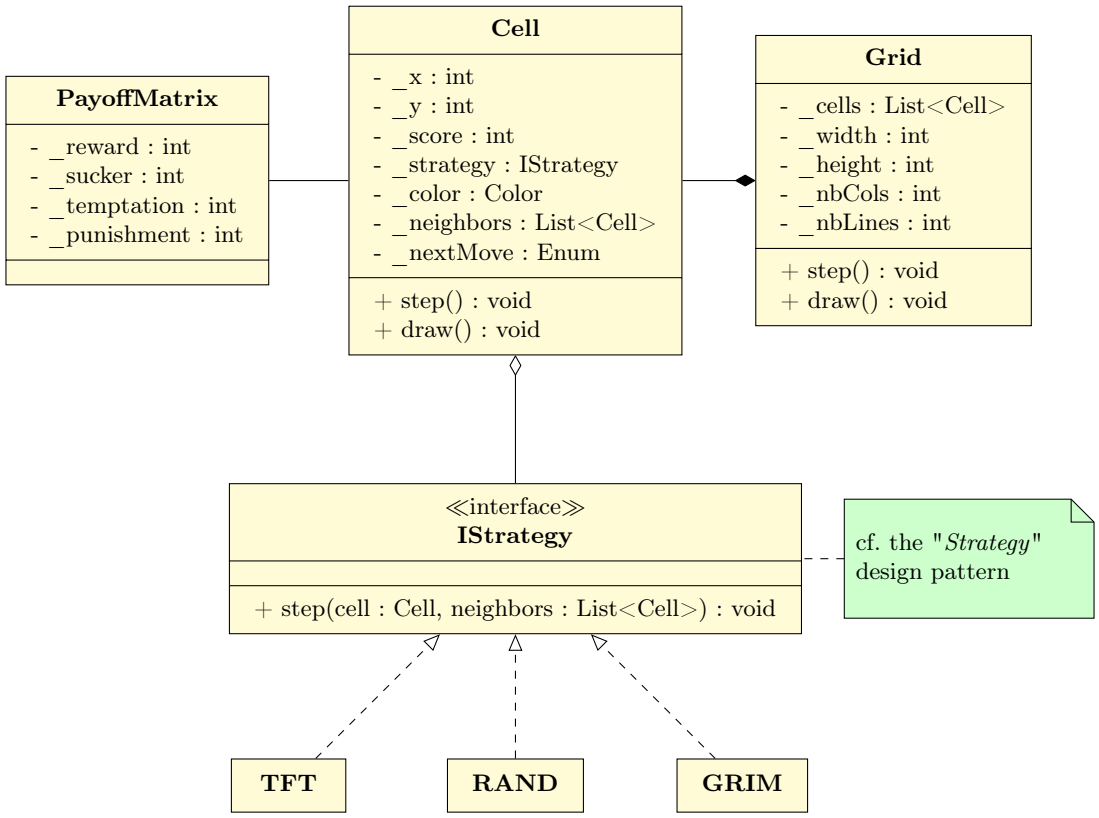

FIGURE 7 – Modèle UML de l'automate cellulaire du dilemme du prisonnier

7.3 Maquette de l'interface

Dans cette partie du document, les diverses interfaces graphiques de l'application seront détaillées et expliquées.

7.3.1 Interaction entre fenêtres

La fenêtre principale de l'application (en bleu) possède deux modes de fonctionnements : Le mode standard et le mode étendu. C'est depuis cette fenêtre que l'on peut accéder au divers menus et fenêtres de l'application.

Dans le cas du schéma ci dessous, on considère la vue principale et la vue étendu comme deux vues différentes. Pour basculer de la vue principale a la vue étendu ou inversement, on actionne un *switch* se trouvant en bas a droite de la fenêtre.

Pour passer de la vue principale (ou étendue) à la fenêtre "à propos", on clique sur le bouton correspondant se trouvant sur la barre de navigation.

Pour passer de la vue principale (ou étendue) à la fenêtre de paramétrage de la matrice des gains, on clique tout d'abord sur l'onglet "*Settings*" de la barre de navigation, puis sur l'option "*Payoff matrix*" du menu déroulant. Idem pour accéder aux paramètres de génération mais en cliquant sur l'option "*Generate new board*" du menu déroulant.

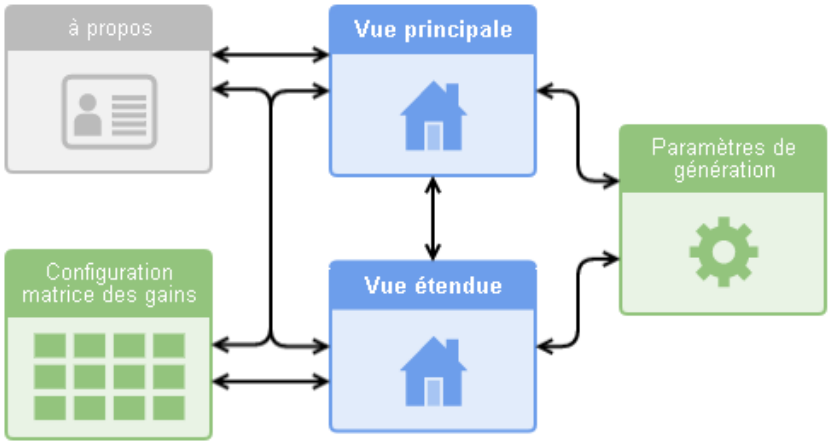

FIGURE 8 – Interactions entre fenêtres de l'application

7.3.2 Fenêtre principale

La fenêtre principale de l'application est composée de plusieurs parties :

Nom du composant	Utilité
La grille :	Composant affichant l'automate cellulaire.
Paramètres de taille :	Permet de modifier le nombre de ligne et colonnes de la grille.
Paramètres de vitesse :	Modifie la vitesse de *step* en mode d'exécution automatique
Bouton *step* :	Passe au temps t_{i+1} de l'automate cellulaire (avance d'un "pas").
Bouton *start / stop* :	Démarre ou arrête l'execution automatique de la commande "*step*"
Bouton *clear* :	Efface le contenu de la grille.
Bouton *extended view* :	Bascule entre la vue principale et la vue étendue.

L'interface suivante est un croquis et il est possible que des fonctionnalités soient rajoutés dans la version finale de l'application.

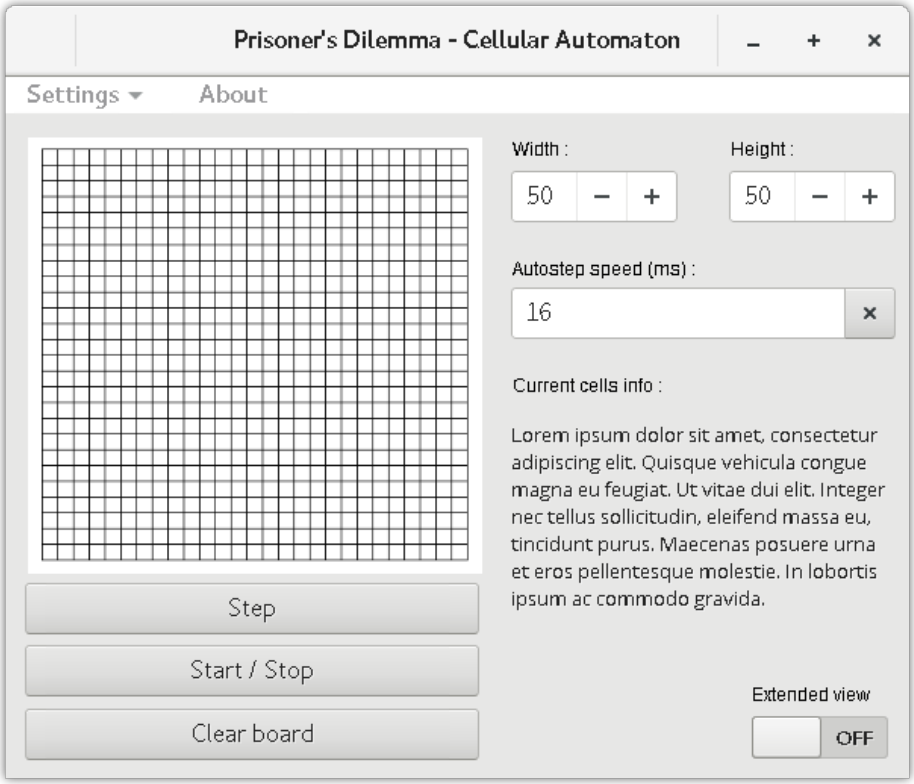

FIGURE 9 – Vue principale de l'application

7.3.3 Fenêtre principale (étendue)

La vue étendue est identique à la vue principale mais possède des graphiques supplémentaires permettant de visualiser plus facilement l'état actuel de l'automate cellulaire.

Voici des exemples de graphiques pouvant être implémentés dans l'application :

— Nombre de cellules "traîtres" par génération.
— Nombre de cellules "coopératives" par génération.
— Pourcentage de chaque stratégies utilisées.
— Stratégie et score maximum associé.
— etc...

Il est possible de basculer à tout moment de la vue étendue à la vue standard en déactivant le *switch* "*extended view*".

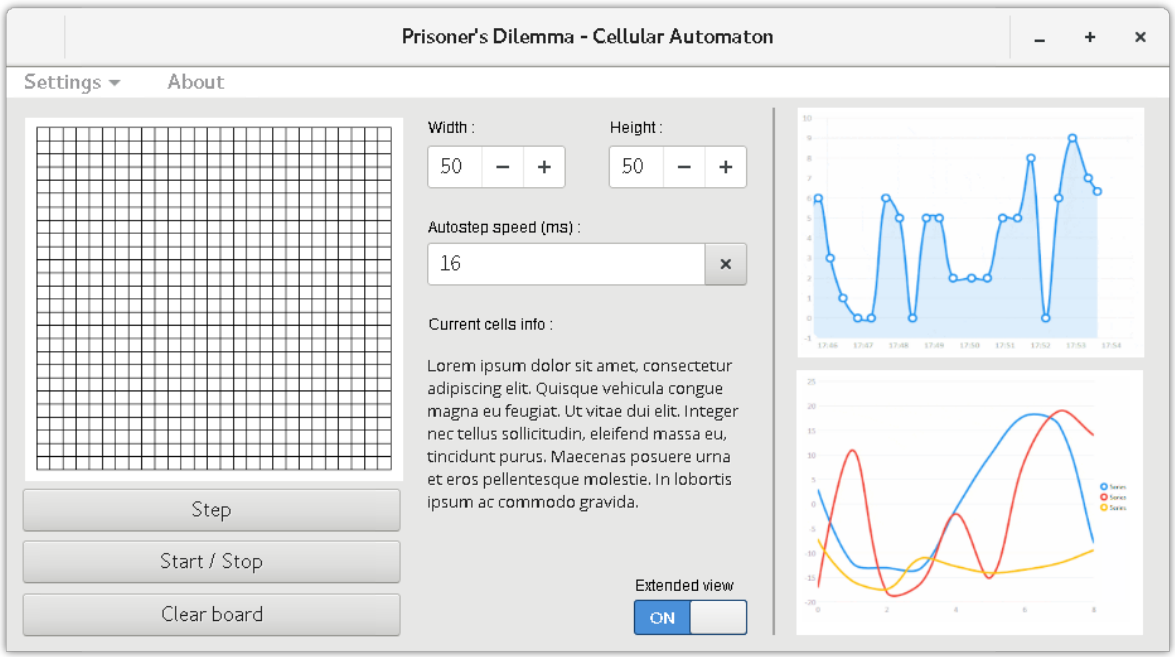

FIGURE 10 – Fenêtre étendue de l'application

7.3.4 Fenêtre principale, paramètres et "à propos"

Sur la fenêtre principale (ou étendue), une barre de navigation est présente en haut de page. Grâce à cette dernière, on peut accéder à un menu déroulant des paramètres de l'application (figure inférieure) et à la fenêtre à propos (figure supérieure).

Depuis le menu déroulant des paramètres, en cliquant sur le bouton "*Payoff matix*", on accède aux paramètres de la matrice de gains (voir "Matrice des gains"). En cliquant sur "*Generate new board*" on accède aux paramètres de la génération d'un nouveau plateau (voir "Paramètres de génération").

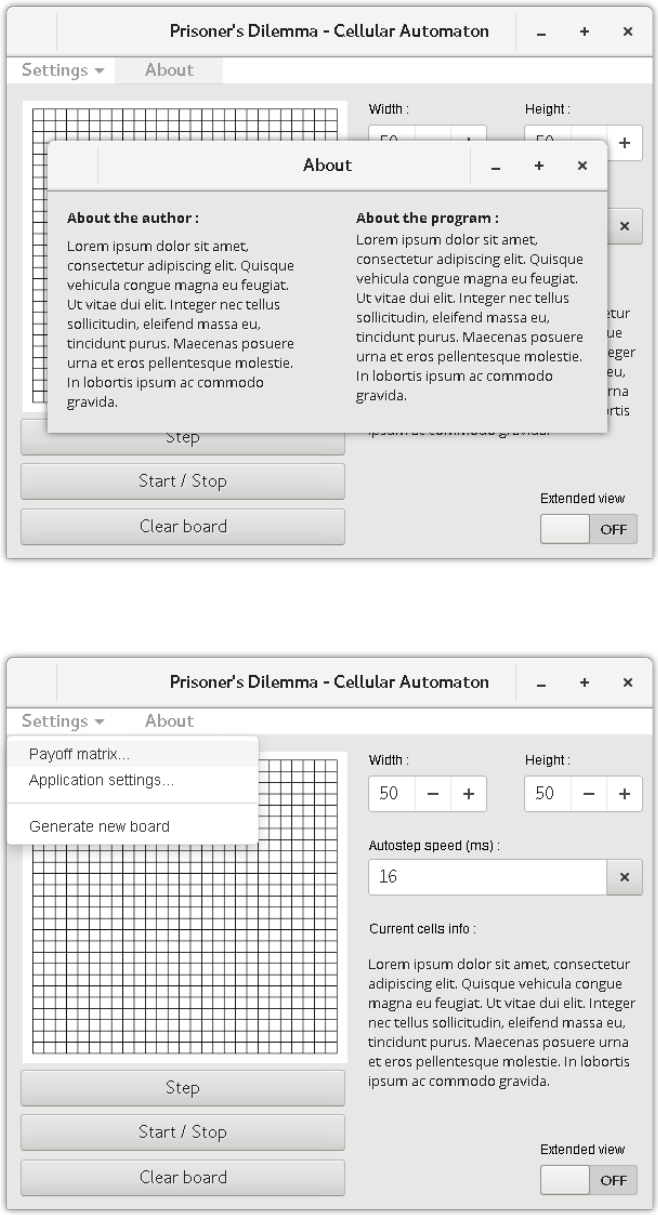

FIGURE 11 – Fenêtre "à propos" et accès aux paramètres de l'application

7.3.5 Matrice des gains

Sur la fenêtre des paramètres de la matrice des gains, on peut modifier différentes valeurs qui par la suite affecteront le comportement des cellules du plateau. Les paramètres présent sur la fenêtre correspondent aux quatre résultats possible lors d'une partie du dilemme du prisonnier.

Les choix sont les suivants :

- *Reward payoff* (R)
- *Sucker's payoff* (S)
- *Temptation's payoff* (T) ou couramment appelé *Cheat's payoff* (C)
- *Punishment's payoff* (P)

On résume donc les valeurs de la matrice par les lettres R pour deux joueurs qui coopèrent, S pour le jouer s'étant fait trahir, T ou C pour le joueur ayant trahi et P pour les deux joueurs s'étant trahi.

On ne peut pas insérer n'importe quelles valeurs dans la matrice des gains. La règle concernant les valeurs de la matrice est la suivante :

$$T > R > P > S$$

En cliquant sur le bouton "*OK*" se trouvant en bas de la fenêtre, on applique les modifications à la matrice des gains et on retourne sur la vue principale (ou étendue). Notez que le bouton "*OK*" de la page sera uniquement activé si les deux conditions citées précédemment sont respectées.

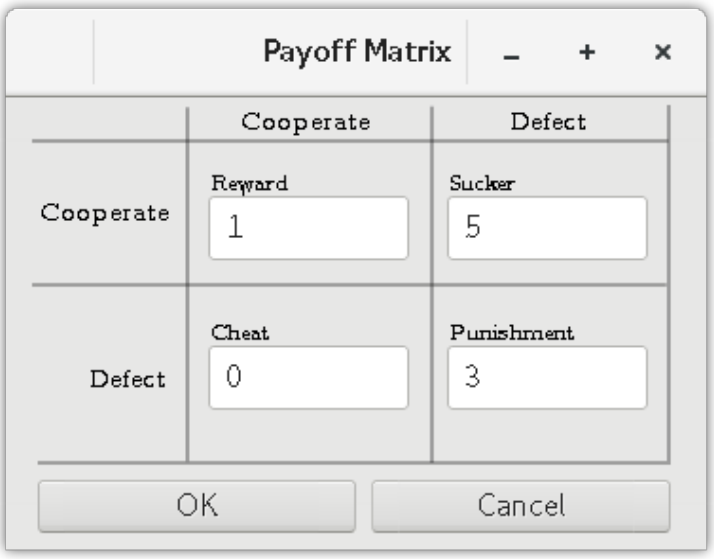

FIGURE 12 – Configuration de la matrice des gains de l'application

7.3.6 Paramètres de génération

La fenêtre "Paramètres de génération" donne la possibilité à l'utilisateur de générer un plateau de cellules avec un répartition aléatoire mais proportionelle des stratégies.

On peut sélectionner jusqu'à quatre stratégies différentes à répartir sur le plateau. Voici un exemple correct de répartition de stratégies :

Random (RAND)	: 15%
Always Defect (AD)	: 15%
Always Cooperate (AC)	: 35%
Grim Trigger (GRIM)	: 35%
Total	: 100%

Le pourcentage total des stratégies sélectionnées doit impérativement être égal à 100%. Si ce n'est pas le cas, l'interface ne permettra pas à l'utilisateur de continuer (voir "Paramètres de génération, contrôles).

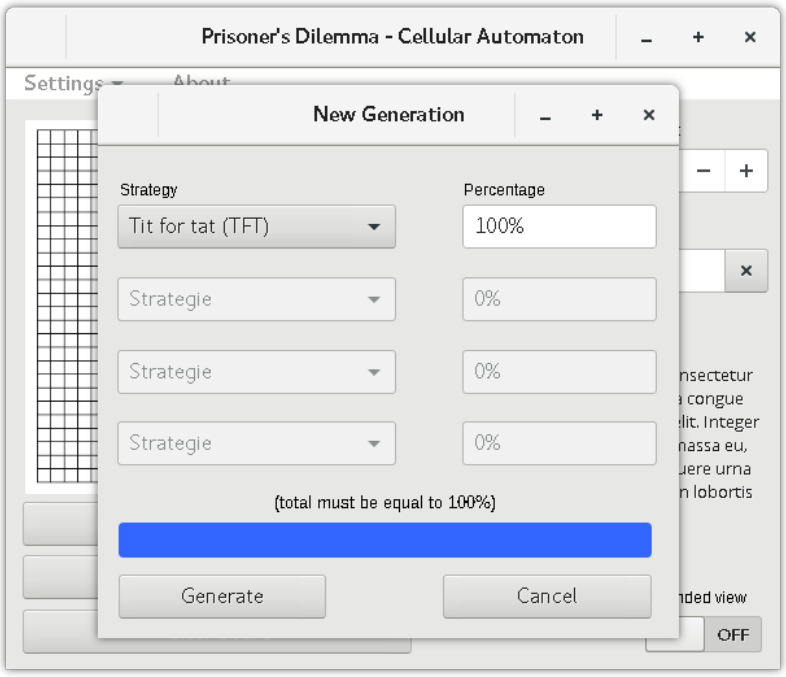

FIGURE 13 – Génération de cellules aléatoirement

7.3.7 Paramètres de génération, contrôles

Cette vue est ici pour démontrer les contrôles de la page "Paramètres de génération" empêchant les utilisateurs de rentrer des valeurs incorrectes. Notez que la barre de progression se trouvant en bas de la page est inférieure à 100%, empêchant ainsi l'accès à génération du nouveau plateau.

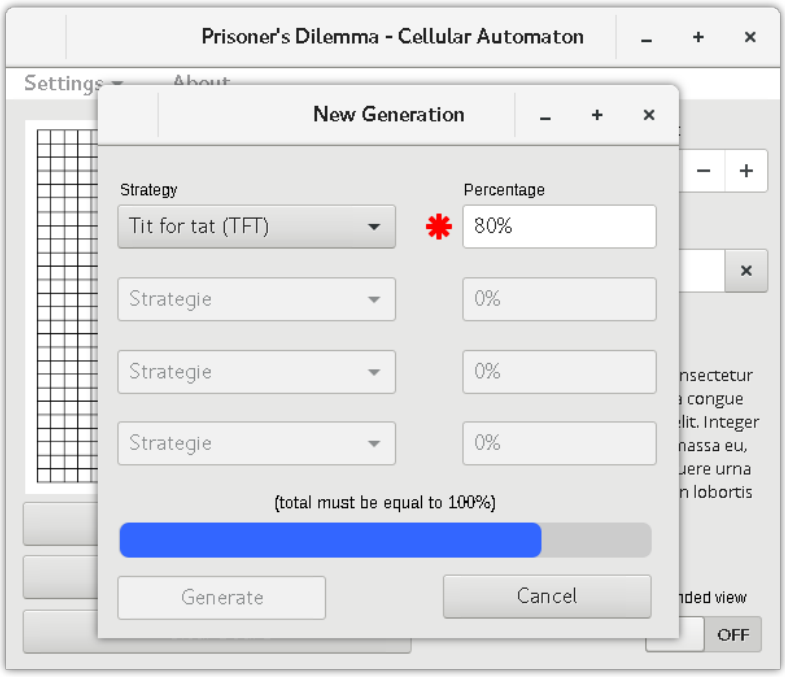

FIGURE 14 – Gestion des erreurs sur la génération aléatoire de cellules

8 Analyse organique

9 Tests

10 Conclusion et perspectives

11 Sources

Références

[1] Investopedia. *Prisoner's Dilemma.*
URL : http://www.investopedia.com/terms/p/prisoners-dilemma.asp.

[2] Eric Weisstein. *Cellular Automaton.*
URL : http://mathworld.wolfram.com/CellularAutomaton.html.

[3] Wikipedia. *Jeu de la vie.*
URL : https://en.wikipedia.org/wiki/Conway's%5C_Game%5C_of%5C_Life.

[4] Wikipedia. *Dilemme du prisonnier.*
URL : https://en.wikipedia.org/wiki/Prisoner's%5C_dilemma.

[5] Wikipedia. *Matrice des gains.*
URL : https://fr.wikipedia.org/wiki/Matrice%5C_des%5C_gains.

[6] Wayne Davis. *Stratégies iterated prisoners dilemma.*
URL : http://www.iterated-prisoners-dilemma.net/prisoners-dilemma-strategies.shtml.

[7] Ramón Alonso-Sanz. *Dilemme du prisonnier, automate cellulaire.*
URL : http://rspa.royalsocietypublishing.org/content/470/2164/20130793.

[8] Livecharts. *Homepage.*
URL : https://lvcharts.net/.

[9] Alonso-Sanz Ramón. *A Quantum Prisoner's Dilemma Cellular Automaton.*
URL : http://rspa.royalsocietypublishing.org/content/royprsa/470/2164/20130793.full.pdf.

[10] Alves Pereira Marcelo. *Prisoner's Dilemma in One-Dimensional Cellular Automata.*
URL : https://arxiv.org/pdf/0708.3520.pdf.

[11] Zbieć Katarzyna. *The Prisoner's Dilemma and The Game of Life.*
URL : logika.uwb.edu.pl/studies/download.php?volid=19&artid=kz.

Table des figures

Liste des tableaux