



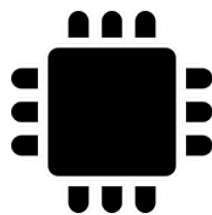
UNIVERSITÉ  
CAEN  
NORMANDIE

Université de Caen Normandie  
UFR des Sciences  
Département Informatique

3ème année de licence d'informatique

# PolTron: La coalition

Expérimentations sur coalition d'IA via le jeu Tron



UFR CAEN

LICENCE INFORMATIQUE  
PROMOTION 2019

Christopher JACQUIOT, Vincent DE MENEZES,  
Alexis MORTELIER, Walid IDOUCHE

Tuteur du projet: Gregory BONNET

Année universitaire : 2018 / 2019

Jury : –  
(si composition du jury connue)

Soutenu le – mars 2019  
(si date connue)

# Table des matières

<b>I. Analyse du projet</b>	<b>1</b>
<b>1. Introduction</b>	<b>2</b>
1.1. Objectif général du projet . . . . .	2
1.2. Objectifs à atteindre . . . . .	2
<b>II. Cahier des charges</b>	<b>3</b>
1.3. Spécifications . . . . .	6
1.3.1. Spécification des paramètres de simulation . . . . .	6
1.3.2. Contraintes techniques . . . . .	6
1.4. Choix techniques . . . . .	6
1.4.1. Schémas de conception . . . . .	6
1.4.2. Langages utilisés . . . . .	7
1.4.3. Accès au code source . . . . .	7
<b>III. Historique des travaux réalisés</b>	<b>8</b>
1.5. Outils de programmation . . . . .	10
1.6. Bibliothèques utilisées . . . . .	10
1.6.1. Module simulation . . . . .	10
1.6.2. Module analyse . . . . .	10
1.7. Interface . . . . .	10
<b>IV. Réalisation</b>	<b>11</b>
<b>2. Simulateur</b>	<b>12</b>
2.1. Nécessités . . . . .	12
2.2. Problème . . . . .	12
2.3. Approches possibles . . . . .	13
2.4. Approche utilisée . . . . .	13
2.5. Remarques sur les résultats obtenus . . . . .	14
2.6. Pistes d'amélioration . . . . .	14
<b>3. Modèle de jeu</b>	<b>15</b>
3.1. Nécessités . . . . .	15
3.2. Problème . . . . .	15
3.3. Approches possibles . . . . .	16
3.4. Approche utilisée . . . . .	16
3.5. Remarques sur les résultats obtenus . . . . .	16
<b>4. Intelligence Artificielle - Heuristique</b>	<b>18</b>

<b>5. Analyse - Exploration</b>	<b>19</b>
5.1. Nécessités . . . . .	19
5.2. Problème . . . . .	19
5.3. Approche utilisée . . . . .	19
5.4. Remarques sur les résultats obtenus . . . . .	20
5.5. Pistes d'amélioration . . . . .	20
 <b>V. Analyse des données générées</b>	 <b>21</b>
<b>6. Analyse des données de l'IA</b>	<b>22</b>
6.1. Analyse d'ensemble . . . . .	22
6.2. Analyses détaillées . . . . .	23
6.2.1. Répartitions des pourcentages de victoires . . . . .	23
6.2.2. Découpage du spectre selon des intervalles de la taille de la carte .	23
6.2.3. Découpage du spectre selon des intervalles de C . . . . .	24
6.2.4. Découpage du spectre selon des intervalles de Ds . . . . .	24
6.2.5. Découpage du spectre selon des intervalles de Dc . . . . .	25
6.2.6. Découpage du spectre selon des intervalles de la différence de niveau	25
6.2.7. Conclusions d'analyse . . . . .	26
 <b>7. Analyse des données du modèle</b>	 <b>27</b>
7.1. Analyse d'ensemble . . . . .	27
7.2. Analyses détaillées . . . . .	28
7.2.1. Répartitions des pourcentages de victoires . . . . .	28
7.2.2. Découpage du spectre selon la taille de la carte . . . . .	28
7.2.3. Découpage du spectre selon des intervalles de C . . . . .	29
7.2.4. Découpage du spectre selon des intervalles de Ds . . . . .	29
7.2.5. Découpage du spectre selon des intervalles de Dc . . . . .	30
7.2.6. Découpage du spectre selon des intervalles de la différence de niveau	30
7.2.7. Conclusions d'analyse . . . . .	31
 <b>VI. Problèmes, tests et expérimentations</b>	 <b>32</b>
<b>8. Problèmes rencontrés</b>	<b>33</b>
<b>9. Conclusion</b>	<b>34</b>
 <b>VII. Annexes</b>	 <b>I</b>
<b>10. Analyse - Simulation</b>	<b>II</b>
10.1. Nécessités . . . . .	II
10.2. Problème . . . . .	II
10.3. Approches possibles . . . . .	II
10.4. Approche utilisée . . . . .	III
10.5. Remarques sur les résultats obtenus . . . . .	V
10.6. Ordre de grandeur de la différence de vitesse de calcul . . . . .	VI
10.6.1. Modèle IA . . . . .	VII
10.6.2. Modèle simulé . . . . .	VIII
10.6.3. Comparaison des deux modèles . . . . .	IX

10.7. Pistes d'amélioration . . . . .	IX
---------------------------------------	----

# Table des figures

10.1. Paramètres initiaux . . . . .	VI
-------------------------------------	----

# Remerciements

Nous tenons à remercier notre tuteur M. Gregory BONNET, pour la proposition de ce sujet passionnant.

## Résumé

Dans ce projet mêlant intelligence artificielle, simulation et analyse, nous allons devoir créer un jeu inspiré de Tron sur lequel nous allons faire jouer plusieurs équipes, une coalition et un joueur seul, leur donnant une différence d'intelligence telle que le joueur solo sera le plus intelligent, et nous allons ensuite devoir analyser les résultats de leurs parties pour déterminer les paramètres les plus optimaux pour que cette coalition gagne contre le joueur seul. Pour réaliser cela nous allons avoir recours à diverses technologies pour résoudre les divers problèmes auxquels nous allons nous confronter. Parmi ces technologies, le python sera utile pour réaliser rapidement notre modèle et notre interface, le Sqlite avec sa portabilité et sa forte intégration avec la plupart des langages sera primordial pour stocker et manipuler les données résultantes de nos simulations, et le langage d'analyse statistique R sera un grand atout pour aider à raisonner rapidement à partir de ces résultats.

## Abstract

In this project about artificial intelligence, simulation and analysis, we will have to make an Tron-inspired game on which we will make two teams fight each other, a coalition and an alone player, both having different intelligence levels, the solo player being the most intelligent, and then analyze the results of their games to determine the optimum parameters to make the coalition win against the solo player. To realize this we will need to make good use of diverse technologies to deal with the problems we will face. Amongst thos technologies, Python will be useful to produce efficiently both our model and interface, Sqlite thanks to it's portability and deep integration with most languages will be primordial to store and manipulate the data resulting from our simulations, and the statistical analysis programming language R will be a great asset to help reason quickly from those results.

*Keywords :* **AI analysis simulation Tron**

**Première partie**

**Analyse du projet**



# 1. Introduction

## 1.1. Objectif général du projet

**Quel est le problème à régler ?** Dans un jeu de Tron dont les règles sont explicitées dans la partie sur le modèle, nous allons faire jouer deux équipes :

- Un joueur seul et intelligent
- Une coalition de joueurs moins intelligents

Le but est d'analyser les meilleurs paramètres pour que notre coalition soit statistiquement la plus efficace contre le joueur seul, si une tendance se dégage de nos simulations.

En d'autres termes, nous allons tenter de répondre à la question :

Combien faut-il d'idiots pour prendre l'avantage sur un joueur plus intelligent ?

## 1.2. Objectifs à atteindre

**Simulateur** Nous allons devoir permettre à la personne voulant générer des données de paramétrer les intervalles et fréquences d'échantillonnage le plus précisément possible pour permettre de générer des données plus précises sur certaines conditions si besoin.

Les paramètres inter-dépendants tels que le nombre de joueurs et la taille des cartes doivent être automatiquement régulés pour un fonctionnement entièrement automatisé.

Une interface permettant de suivre la progression de la simulation est aussi très importante pour estimer quand terminent nos simulations.

### Modèle de jeu

### IA et son heuristique

**Stockage de masse** Au vu des grandes quantités de données potentielles, une base de donnée bien structurée avec des vues permettant de faciliter l'accès aux informations pertinentes pour l'analyse sera primordiale.

**Analyse statistique** Nous allons devoir faciliter la visualisation et le travail sur nos données afin de permettre de se concentrer sur l'analyse plutôt que sur les outils d'analyse. Il sera donc important d'unifier au possible les moyens d'analyse des données et de rédaction d'analyses pour augmenter notre efficacité.

**Deuxième partie**

**Cahier des charges**

Fonction de service	Critère/Module	Niveau	Flexibilité	Contributeur au module
FONCTION PRINCIPALE				
FP 0	Exemple d'utilisation	Expliquer comment utiliser le cahier des charges	Chaque étudiants doit comprendre comment utilisé le cahier des charges; <del>Une fois une tâche réalisée celle-ci doit être rayé par un code couleur (rouge : fait mais à optimiser/ou pas sûr (souvent le cas)/vert : finis parfaitement;</del> Une fois tous les éléments de la case flexibilité rayées ("rouge"/"vert") le module et la fonction de service doivent être mis en vert (si le module doit être abandonné pour des raisons x ou y celui-ci doit être mis en rouge);	Vincent
FP 1	Jeu Tron	Création d'un plateau, avec les règles du jeu Tron afin de créer un "jeu" jouable avec plusieurs controleur	Déplacement sur le plateau avec des commandes simples; Laisse un mur à la position d'origine lors d'un déplacement; Déplacement possède sens 4 directions possible; Partie finis lorsqu'il ne reste plus de joueur dans les 2 camps ("bleu"/"rouge"); Un joueur est éliminé lorsqu'il se trouve sur une case de type mur; Type de case ("mur"/"vide"); Extrémité du plateau composé de case de type ("mur"); Un tour d'un joueur est composé d'un seul déplacement obligatoire avec 3 directions possible; Dans un cycle de tour chaque joueurs jouent 1 tour; Implémentation d'un compteur de tour;	Alexis
FP 2	Intelligence Artificielle	Choisis un déplacement le moins risqué déterminer par l'analyse du plateau	Implémentation de l'algorithme paranoïde; <del>Heuristique : vérification de la zone de contrôle d'un camps sur le plateau;</del> Stockage/lecture des états d'un graphe;	Vincent
FP 3	Analyse des expérimentations	Création d'un algorithme permettant d'établir un graphe de chaleur/autre, à partir de plusieurs résultats.	Le résultat d'une simulation sera composé de plusieurs informations ("taille/aire du plateau", "nombre de la coalition", "profondeur de recherche des 2 camps", "moyenne de victoire contre la coalition (en %)", "position de départ****", "ordre des joueurs****"); "le nombre de tour"); Avec les différentes informations d'une simulation, créer un système permettant de juger une partie selon le taux de victoire + le nombre de tour, afin de déterminer une seule position dans le graphe de chaleur; Faire une moyenne de victoire lorsqu'il existe une configuration identique avec un résultat différent; ****fonctionnalité aléatoire ne doit pas être systématiquement pris en compte pour commencer les premières expériences	Christopher
FP4	Simulateur	Génère l'espace de recherche (paramètres initiaux) et gère l'ajout des résultats de chaque partie à la bdd	Le simulateur doit générer toutes les combinaisons valides et intéressantes de paramètres initiaux (M,N, C, Ds, Dc) et lancer un nombre suffisamment grand de parties à ordre de joueur et positions initiales aléatoires avec ces parametres, pour explorer les statistiques de victoires de chacunes de ces combinaisons	Christopher
FP 5	Rapport	Présenter le travail du groupe	Expliquer les modules implémenter; Montrer les résultats obtenus; Expliquer/Faire des hypothèses avec les résultats obtenue;	
FONCTION DE CONTRAINTE				

FC 1	Jeu Tron	L'initialisation du plateau doit être facilement modifiable	Position des joueurs; Taille du plateau;	
FC 2	Intelligence Artificielle	Elasticité des recherches en profondeur	Profondeur des 2 camps facilement configurable;	
FC 3	Intelligence Artificielle	Stockage de plateau	Le Stockage doit avoir plusieurs états, pouvant être identifier à un plateau et fournir une direction qui a déjà été calculé; Il sera composé de l'état + la direction à choisir;	
FC 4	Jeu Tron	Stockage d'une partie	Le Stockage d'une partie doit garder les configurations des parties avec le nombre de parties simulé suivis du taux de réussite de la partie;	
FC 5	Rapport	Rédaction	Le rapport devra être rédigé en LATEX	
FC 6	Module	Autonomie	Chaque modules doivent pouvoir se débrouiller seul sans l'aide d'autre module; Le partage d'information se fera à l'aide du stockage; Permet une diversification dans les langages à adopter;	
FC 7	Logiciel	Langage	Chaque modules doivent avoir un langage qui convient pour sa taches à effectuer; Le langage doit pouvoir être facile à manipuler pour les modules;	
FC 8	Jeu Tron	Taille du plateau	Le plateau peut être rectangulaire	

---

## 1.3. Spécifications

### 1.3.1. Spécification des paramètres de simulation

**Attention** Pour pouvoir simuler correctement, il est nécessaire que les paramètres d'entrée donnés au simulateur ne brisent pas les règles suivantes.

$$param > 0 \quad (1.1)$$

$$\min_X \leq \max_X \quad (1.2)$$

$$0 < Dc < Ds \quad (1.3)$$

### 1.3.2. Contraintes techniques

**Temps imparti réduit** Suite à une annulation de la matière puis à la réouverture de celle ci, le temps imparti pour ce projet as été considérablement amoindri.

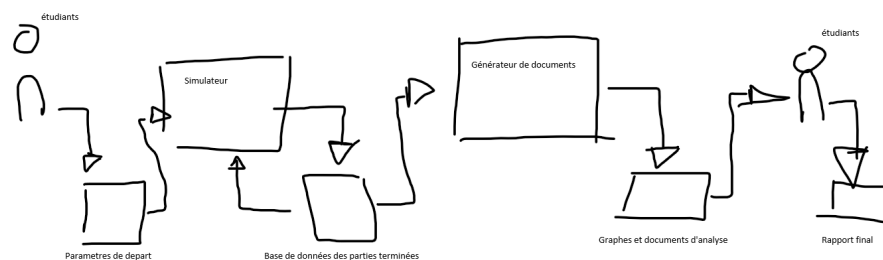
Nous avons environ 5 semaines pour mener ce projet à terme à compter du 31 janvier 2019. Il est donc nécessaire de réduire un maximum les temps de développement pour le mener à bien.

Cela a mené à la nécessité d'évaluer nos options de façon la plus pragmatique possible en termes de coûts en temps d'implémentation.

## 1.4. Choix techniques

### 1.4.1. Schémas de conception

#### Architecture



#### Base de données

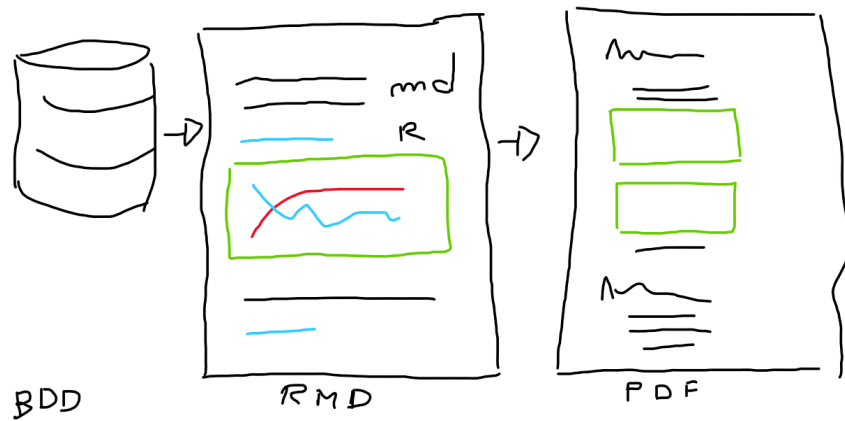
game:	player_order:	initial_positions:	Game_results:	Important moments:	Deaths:
*partie_id	*partie_id	*partie_id	*partie_id	*partie_id	*partie_id
M	Ini_Pos	Ini_Pos	Ended_tick	tick	Tick
N	Player_Id	Player_Id	Won	Plateau_string	Player_id
Ds	Order	Order		N_walls	
Dc				Coalition_deaths	
C					

---

**Modèle de jeu**

**Heuristique**

**Analyse**



#### 1.4.2. Langages utilisés

**Module simulation :**

- Python pour l'interface commande et les sous modules internes.
- SQL pour la génération et l'interaction avec la bdd

**Module analyse :**

- R pour la génération des graphes et la manipulation des données
- Markdown pour la rédaction du rapport d'analyse

#### 1.4.3. Accès au code source

Vous pouvez trouver [l'intégralité du code source ici](#).

**Troisième partie**

**Historique des travaux réalisés**

DATE	PRENOM	MODULE	COMMENTAIRE	TEMPS	RESTE A FAIRE	PROBLEME (FACULTATIF)
31/01/2019	Vincent	Mise en place : creation du drive et des fichiers commun pour le déroulement du projet	création d'un drive + planning + cahier des charges	1H	Accepter les autres membres et configurer le Drive	N/A
31/01/2019	Christopher	Mise en place et analyse préliminaire	rédaction du récapitulatif du projet + analyse des objectifs du projet + analyse d'une optimisation algorithmique pour evaluation de minmax	2H	analyse du format des données en externe et interne	N/A
31/01/2019	Christopher	analyse préliminaire	analyse des données à stocker + analyse des données de la simulation + analyse de l'architecture du projet	2H	analyse des algorithmes internes + analyse des moyens présents + analyse des outils utilisés	N/A
31/01/2019	Christopher	Préparation	Ajout d'un résumé de mes expériences sur les outils potentiels et de mes potentiels outils déjà prêts	1H	N/A	N/A
01/02/2019	Vincent	Cahier des charges	Création du cahier des charges fonctionnel	3H	Compléter les informations manquantes (si présente)	N/A
01/02/2019	Vincent	Mise en place: config du Drive	Ajout des étudiants et des droits	-	N/A	N/A
01/02/2019	Vincent	Récapitulatif	Section Expériences (Vincent) remplis	30MIN	N/A	N/A
01/02/2019	Alexis	Analyse du projet	Lecture de différentes doc sur le jeu, les différentes IA, exemple de code, amélioration possible, infos importante au "n" moment pour avoir un graphe intéressant, etc.	3H	N/A	N/A
01/02/2019	Walid	Analyse du projet	Analyse des documents du jeux , essayer de comprendre le fonctionnement du projet .	2H	N/A	N/A
02/02/2019	Christopher	Création du module de bdd	Design et génération automatique de la bdd + interface python d'entrée des données	2H30	Test avec de vraies données et potentiellement ajout du stockage du cache pour l'heuristique de l'IA	Nécessite d'avoir l'heuristique de prêter avant de pouvoir déterminer le format du cache
02/02/2019	Christopher	Création d'un prototype d'analyser	Générateur de données aléatoire et rendu des graphes fonctionnels	6H	Avoir de vraies données pour affiner les affichages les plus intéressants + implémenter des graphes de comparaison de profondeur de recherche	N/A
02/02/2019	Vincent	Conversion heuristique	De JAVA à Python	6H	Rendre l'utilisation de l'heuristique compatible avec la structure du jeu	Reconnaissance des camps bleu et rouge non automatisé
03/02/2019	Christopher	Amélioration Analyses	Ajout d'une méthodologie d'exploration statistique et des fonctions permettant la génération des graphes utiles	6H	Dupliquer les analyses pour chaque facteur d'entrée des parties et chaque élément d'analyse voulu	N/A
02/02/2019 03/02/2019	Alexis	Conception du jeu en mode joueur contre joueur	Implementation de toutes les fonctionnalités et de l'architecture du jeu Tron*	10H	déplacements	N/A
03/02/2019	Christopher	Couche de gestion de la simulation	Génère et teste autant de fois que voulu des parties selon toutes les combinaisons de paramètres de départ voulues, gère l'envoi à la bdd et l'affichage du temps restant estimé	8H	Brancher la simulation sur de vraies parties	En attente du modèle de partie pour le reste
04/02/2019	Christopher	Interface utilisateur et valeurs par défaut	Permet de modifier les paramètres de départ et d'en savoir plus sur les arguments de recherche	2H	N/A	N/A
04/02/2019	Christopher	Début de rédaction du rapport	Retraits de catégories inutiles pour notre projet, ajout de début de contenu sur les outils utilisés, préparés pages de chapitres potentiels, ai rédigé le résumé et l'abstract du projet	3H	Rédiger le contenu des chapitres sur ce que j'ai fait; mettre à jour les pdf de cahier des charges quand le projet sera terminé	Il faudra remettre à jour les pdf de cahier et d'historique à la fin du projet
04/02/2019	Christopher	Début de rédaction du rapport	Ajout de schémas et des documents internes (cahier, historique, analyse); Début de rédaction de titres de paragraphes et de sections dans mes chapitres	3H	Continuer de rédiger	voir au dessus
04/02/2019	Alexis	Finition des déplacements, et optimisation primaire du code	Voir dans le détails s'il n'y a pas possibilité d'optimisé encore plus	2H	Implementation ressortant les informations importantes à l'analyse du jeu	N/A
05/02/2019 06/02/2019	Alexis	Changement de pensée du code du jeu, optimisation de celui-ci	Code optimisé à fond, 60 fois plus rapide qu'avant	4H	FIIIIIIIIIIIEEE	N/A
06/02/2019	Christopher	Compilation Cython	Compilation fonctionnelle du code python via cython et application d'optimisations pour la transcompilation python -> C	2H	N/A	N/A
06/02/2019	Christopher	Modélisation du système	Création d'une modélisation basée sur de la physique pour tenter de simuler notre intelligence d'une façon différente	3H	Tweaker le modèle quand on aura accès aux données de l'IA pour tenter de le faire fit le plus possible.	N/A
06/02/2019	Christopher	Debug de la toolchain Simu - > db -> rapport complet	Debugging complet de l'insertion des données dans la bdd et de sa lecture par R	1H	N/A	N/A
07/02/2019	Christopher	Optimisation du modèle et des données récoltées	Retrait de la string d'état pour accélérer un maximum la génération de données importantes et éviter les opérations inutiles, gain de temps final sur l'exécution du modèle: ~25%	2H	N/A	N/A
07/02/2019	Christopher	Écriture du chapitre sur la simulation de modèle	Écriture complète et production de schémas illustratifs pour le chapitre de la simulation.	4H	N/A	Ajout du pdf d'analyses du modèle
07/02/2019	Christopher	Écriture du chapitre sur le simulateur	Écriture complète et production de schémas illustratifs pour le chapitre du simulateur.	4H	N/A	N/A
08/02/2019	Christopher	Ajout de facteurs et amélioration du rendu des analyses	Ajout des facteurs M*N et Ds-Dc à nos analyses de corrélations	2H	N/A	N/A
08/02/2019	Christopher	Écriture du chapitre sur l'exploration des données	Écriture complète et production de schémas illustratifs pour le chapitre sur l'exploration statistique.	3H	N/A	N/A
08/02/2019	Christopher	Rectifications d'après retours	Ajouté plus de détails dans l'abstract concernant le projet lui-même, corrigé des fautes oubliées, modifié légèrement quelques phrases grammaticalement incorrectes	10MIN	N/A	N/A
04/02/2019 08/02/2019	Chris's PC	Génération d'huile de coude (simulation de modèles)	Génération de données résultant d'un total de 2-3 millions de parties simulée via notre modèle physique sur différents sets de données	~40H	Recommencer pour le modèle avec IA	R.I.P. PC, bientôt tu pourras te reposer x)
08/02/2019	Christopher	Analyse et rédaction des résultats des données du modèle	Écriture complète et production de schémas illustratifs pour le chapitre d'analyse des résultats du modèle	4H	voir ci-dessus	N/A
09/02/2019	Vincent	Implémentation Heuristique avec le jeu	Implémentation en objet et non objet d'une heuristique, celle-ci calcule la portée de tout les joueurs, puis attribue la case la plus proche aux équipes. Ajout d'une méthode permettant la vérification des portées de chaque joueur.	9H	Faire des tests avec une simulation plus grande (nombre de la coalition, taille du plateau); Légère amélioration des conditions dans la mesure du possible	N/A
10/02/2019	Christopher	Calcul de différence de vitesse entre simulation et modèle	Calcul et ajout de la différence du nombre de cases à checker sur la même partie extrême entre le modèle simulé et le modèle d'IA	2H	N/A	N/A
10/02/2019	Vincent	Algorithme Paranoïde	Création de 2 versions d'algorithme paranoïde; Optimisation de l'algorithme paranoïde	13H	Optimiser le code; Changer les noms de variables/fonctions (nettoyer le code)	Beaucoup de partie ne peuvent pas être jouées, car l'IA ne trouve pas de coup optimal. Solution: une nouvelle heuristique/ vérifier la fiabilité de l'algorithme
11/02/2019	Vincent	Recherche algorithme	Phénomène de prédiction (profondeur plus grande que son adversaire)	5H	N/A	L'IA se retrouve à jouer un coup légal, dû à l'utilisation de l'algorithme minimax
12/02/2019	Vincent	Recherche algorithme	Recherche d'un algorithme permettant de jouer une partie de Tron	5H	Trouver un algorithme capable d'accepter l'heuristique prévue, ou trouver une heuristique compatible avec l'algorithme	L'heuristique renvoie la taille zone, suivant qui l'exécute, le mode de jeu devient "un chasseur qui chasse sa proie, et la proie qui fuit".
15/02/2019 16/02/2019	Christopher	Refonte totale du système de jeu et de l'intégration à l'IA	Optimisation algorithmique et refonte du système de jeu suite à un oubli de conception (ordre aléatoire), Remplacement du système de copie de l'IA par un système de rollback d'actions plus lisible et efficace. Optimisation de l'heuristique.	12H	N/A	N/A
16/02/2019	Christopher	Analyse des données résultant des simulations IA et Modèle	Analyse des statistiques résultant des données générées par le simulateur. Réécriture des analyses du modèle pour coller aux nouvelles données générées par les mêmes options que pour la simu d'IA.	5H	N/A	N/A
16/02/2019	Chris's PC	Génération de données	Détermination de paramètres de simulation faisables manuellement, et génération sur 2h30 des données par simulation d'IA et du modèle physique	3H	N/A	N/A



---

**Concernant Walid** Suite à une discussion sur les expériences, et compétences de chacun pour analyser comment mener au mieux ce projet, un accord a été passé avec Walid pour qu'il puisse se familiariser de son côté avec Python et aux concepts du projets en tentant d'en réaliser un maximum de son coté.

Afin de ne pas le délaissier non plus, il a été encouragé à poser ses éventuelles questions et à s'inspirer du code principal pour expérimenter et rattrapper son éventuel retard sur certains concepts.

## 1.5. Outils de programmation

**Alexis :**

**Vincent :**

**Christopher :**

- Pycharm + l'extension Sonar Lint pour programmer en Python
- Rstudio pour programmer le projet en R et étudier le contenu de la base de données

**Walid :**

## 1.6. Bibliothèques utilisées

### 1.6.1. Module simulation

- time pour estimer le temps restant avant completion des simulations
- sqlite3 pour l'interfacage avec la bdd sqlite

### 1.6.2. Module analyse

- dplyr pour faciliter la manipulation et la selection par sémantique des données
- ggplot2 pour ses graphes de qualité et facile à configurer
- GGally pour ses outils d'analyse de tables de données complètes
- RSQLite pour l'interfacage avec la bdd sqlite

## 1.7. Interface

```
Total amount of simulations to do: 80000
highest map M size: 50      sampled every 10
highest map N size: 50      sampled every 10
highest Coalition size: 6    sampled every 5
highest solo research level: 10 sampled every 1
highest coalition research level: 9 sampled every 2

Time estimated 0d 0h 20m 46s |-----| 0.2% @ 0.015625s/game
```

## **Quatrième partie**

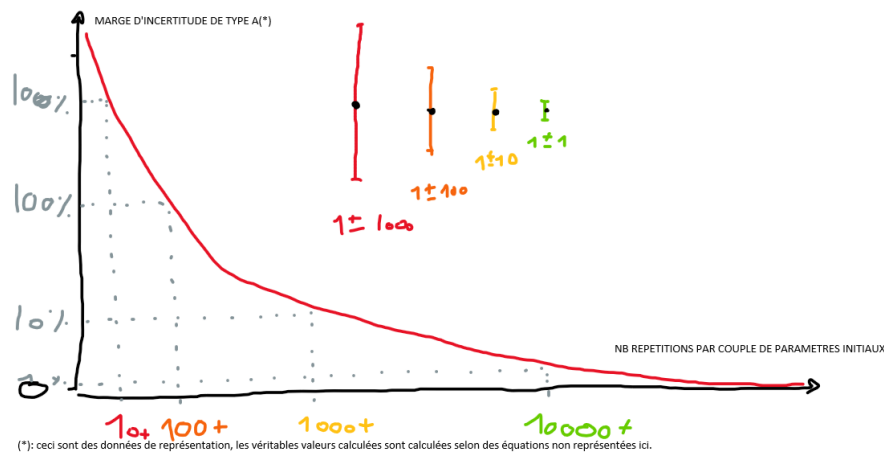
### **Réalisation**

## 2. Simulateur

### 2.1. Nécessités

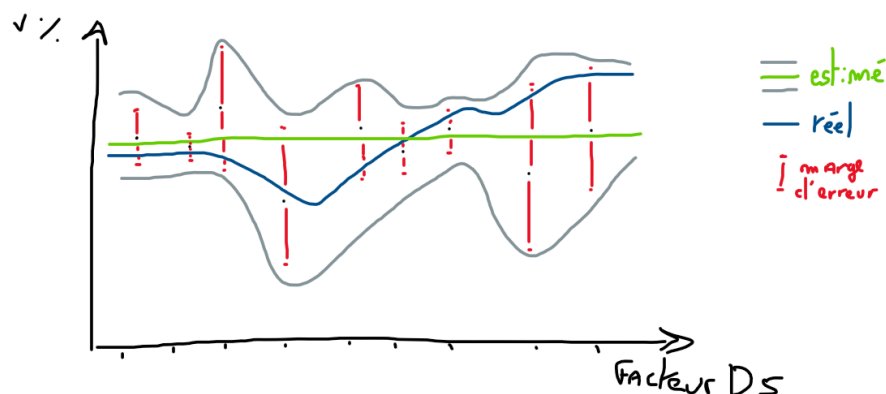
**Tester de façon uniforme notre espace de recherche** Afin de pouvoir avoir des statistiques les moins biaisées possible, il est nécessaire d'uniformiser nos simulations sur notre espace de recherche afin d'éviter la sous-représentation de certains couples de paramètres initiaux.

### 2.2. Problème



**Comment maximiser la précision statistique d'un couple de paramètre unique ?** Pour déterminer le pourcentage de victoires d'un certain couple de paramètres initiaux, nous allons avoir besoin de réaliser un certain nombre de simulations.

Cependant, quelques tentatives ne seraient potentiellement pas représentatif du pourcentage de victoire, un peu comme 3 lancers de pièces ne font pas 50-50 % de chances d'avoir pile ou face. Il nous faudrait donc répéter notre simulation un maximum de fois pour déterminer l'incertitude statistique de notre mesure. Mais combien de fois ?



**Comment éviter des erreurs d'estimations statistiques ?** Avec des données mal réparties, nous pourrions avoir des soucis d'estimations. Le graphique ci-dessus est un exemple de mauvaise représentation potentielle, dû à la différence de marge d'erreur.

Des données avec les mêmes marges d'erreurs auraient pu potentiellement au moins retrouver le premier creux en essayant de coller un maximum les points. Dans le cas illustré, le calcul pourrait avoir considéré le point haut en incertitude du creux comme une anomalie comparée aux autres points relativement alignés, résultant en une estimation faussée de la forme de nos données.

Évidemment plus de données est toujours mieux pour réduire la marge d'erreur générale de notre estimation, mais comment éviter au moins un maximum cette déformation ?

Comment pourrions nous maximiser la précision de nos analyses et la lisibilité de nos résultats ?

### 2.3. Approches possibles

**Génération aléatoire et uniforme de paramètres initiaux** Nous pourrions tirer parti de l'aléatoire pour générer de façon aléatoire mais uniformément des couples de paramètres initiaux.

Cela aurait le mérite de pouvoir avoir une image globalement représentative de notre phénomène avec de moins en moins de déformations dues à l'aléatoire à mesure que nous multiplions le nombre de tirage au sort de paramètres.

Le souci avec cette approche est que nous pourrions potentiellement subir les aléas d'un générateur pseudo aléatoire pas réellement uniforme qui pourrait biaiser nos résultats, et que selon notre espace de recherches, il serait nécessaire d'avoir beaucoup de tirages au sorts pour s'assurer de la précision sur certaines données.

**Génération complète des points de l'espace de recherche** L'approche inverse serait de générer exactement toutes les combinaisons possibles de paramètres initiaux de notre espace de recherche, et de les répéter un nombre suffisant de fois pour satisfaire le niveau de précision voulu sur chacun de ces points.

La précision de cette approche serait alors directement liée au nombre d'itérations par combinaisons mais aussi potentiellement plus gourmande en simulations que l'approche aléatoire.

### 2.4. Approche utilisée

**Exploration complète d'un espace de recherche voulu**

Nous avons préféré partir sur un simulateur parcourant l'intégralité de notre espace de recherche pour minimiser les biais et aléas d'un générateur aléatoire et ainsi maximiser la précision de nos résultats.

La grande quantité de simulations combinée à la vitesse de calcul du déroulement d'une partie peuvent vite faire durer le processus de génération de données sur plusieurs minutes à plusieurs heures selon l'espace de recherche, mais les données en résultant sont les plus fidèles que nous pourrions avoir en un minimum de temps de génération.

## 2.5. Remarques sur les résultats obtenus

```
Total amount of simulations to do: 657500
highest map M size: 50      sampled every 5
highest map N size: 50      sampled every 5
highest Coalition size: 40   sampled every 5
highest solo research level: 10 sampled every 2
highest coalition research level: 9 sampled every 2

Time estimated 0d 20h 46m 17s |-----| 8.3% @ 0.124s/game curr (45,35,21,8,1) sim#54268
```

**Les performances du modèle de simulation sont critiques** La grande quantité de simulations nécessaire pour évaluer un espace de recherche à 5 dimensions sur de petits intervalles à une précision convenable rendent le temps d'exécution des simulations cruciales pour générer nos données en un temps raisonnable.

```
ncalls tottime percall cumtime percall filename:lineno(function)
1 0.000 0.000 0.204 0.204 <string>:1(<module>)
2 0.000 0.000 0.000 0.000 abc.py:137(__instancecheck__)
178122/6896 0.132 0.000 0.177 0.000 model.py:116(propagate_case_search)
719 0.000 0.000 0.000 0.000 model.py:131(has_ended)
```

Notre langage de départ étant Python, nous avons optimisé notre vitesse d'exécution à l'aide du transcompilateur Cython qui permet de générer du code C à partir de code source Python.

Pour accélérer encore plus nous avons tiré parti de la capacité de Python à intégrer du typage statique via les annotations pour indiquer à Cython les types des variables et le laisser optimiser encore plus profondément les algorithmes C utilisés, en plus d'avoir des indications plus complètes et lisibles pour la documentation en bonus.

```
ncalls tottime percall cumtime percall filename:lineno(function)
1 0.085 0.085 0.085 0.085 <string>:1(<module>)
2 0.000 0.000 0.000 0.000 abc.py:137(__instancecheck__)
41 0.000 0.000 0.000 0.000 random.py:224(_randbelow)
1 0.000 0.000 0.000 0.000 random.py:264(shuffle)
1 0.000 0.000 0.000 0.000 random.py:286(sample)
```

**Les données sont bel et bien réparties de façon uniforme** Grâce à cette approche, nous pouvons bel et bien voir l'uniformité de nos tests sur les paramètres initiaux, la taille maximale de la coalition étant considérée variable selon la taille du plateau, il est cependant normal de voir une densité plus forte de tests plus M et N grandissent, conformément à la taille supérieure de l'intervalles de valeurs C à tester sur ces dimensions d'arène.

Mais même cette augmentation de densité est uniforme. Nous pouvons retrouver ce genre d'informations sur les graphes de densités de nos analyses.

## 2.6. Pistes d'amélioration

**Simulations en parallèle** Nous avons tenté de faire de multiples simulations en parallèle pour pouvoir profiter des multiples coeurs de nos systèmes de calculs, mais notre Cython a malheureusement souffert de l'overhead Python de la librairie multiprocessing et l'exécution s'est révélée plus lente que sans.

Une implémentation du multiprocessing directement en C ou via une librairie Python déjà optimisée pour Cython devrait permettre d'accélérer grandement les calculs de simulation en parallélisant la charge de calcul sur autant de coeurs que possible.

## 3. Modèle de jeu

### 3.1. Nécessités

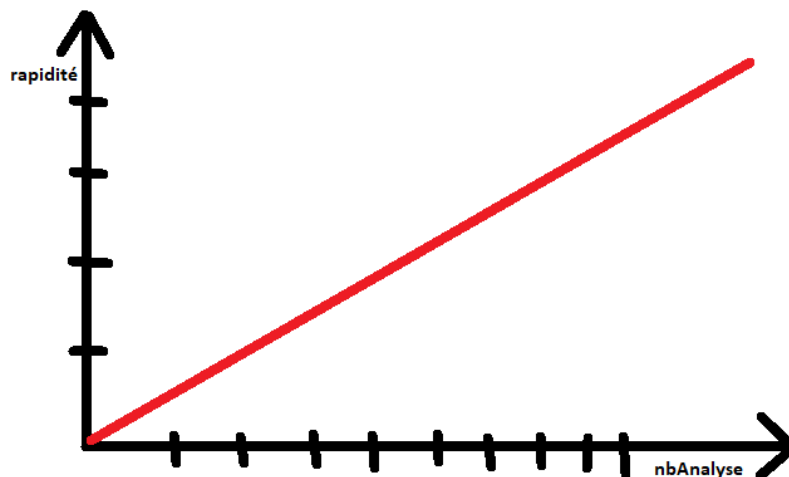
**Un moteur de jeu efficace** Dans le cadre de notre projet tutoré nous devons répondre à la problématique : "Combien faut-il d'idiots pour prendre l'avantage sur un joueur plus intelligent ?". Pour y répondre, nous avons besoin d'analyser beaucoup de parties différentes.

Pour améliorer les performances générales, nous avons opté pour une transcompilation python vers C en profitant des optimisations apportées par Cython grâce à son support des annotations de types. Mais cela n'est pas sans coût.

### 3.2. Problème

**Comment rendre le moteur efficace ?** Afin de pouvoir maximiser la fiabilité de nos analyses, nous allons avoir besoin d'en réaliser un maximum. Cela dit, pour notre projet nous disposons d'un temps limité pour les réaliser.

Comment pourrions nous rendre le moteur le plus efficace possible ?



**Comment implémenter le moteur efficacement ?** Malheureusement, ce temps limité nous incombe aussi de devoir réaliser ce moteur au plus vite pour lancer les simulations au plus tôt.

Comment pourrions nous implémenter ce moteur le plus efficacement possible ?

Comment pourrions nous rendre notre moteur le plus rapide possible pour simuler un maximum d'analyses en un temps imparti en minimisant le temps d'implémentation ?

### 3.3. Approches possibles

**Approche Programmation Orientée Objet** La POO as l'avantage d'être modulaire, abstraite et aisément réutilisable, permettant une implémentation très naturelle et rendant le code très compréhensible, diminuant les sources potentielles de bugs et accélérant le développement du moteur.

Cependant son aspect pratique se paye par son économie en ressource. Certaines fonctionnalités des classes python ne sont pas véritablement gérées par cython et nécessitent une évaluation de code python classique, ajoutant un overhead à l'exécution de leur code par cython. De ce fait, utiliser pleinement les classes python peut ralentir l'exécution finale du programme après transcompilation.

**Approche structures de bases** Au lieu de créer des classes wrapper, il est aussi possible de programmer nos structures à partir d'un maximum de structures de base, plus spécifiques, concises, et sans overhead, mais sans compartimenter les systèmes dans des sous systèmes dédiés, les sources d'erreurs et de bugs peuvent augmenter, et ralentir l'implémentation du moteur.

Mais cela permet de bénéficier d'un maximum de gains de performances de la part de Cython.

### 3.4. Approche utilisée

**Approche finalement choisie** Nous avons finalement choisi un mix des deux options en minimisant au possible le nombre de classes. Les gains de performances apportés par Cython sont actuellement d'une exécution 5 fois plus rapide en moyenne sur des parties de paramètres ( $M=10$ ,  $N=10$ ,  $C=4$ ,  $D_s=4$ ,  $D_c=3$ ).

Et le code reste malgré tout un maximum lisible et clair.

Minimiser l'usage de classes à permis d'accélérer l'exécution moyenne de notre moteur d'un facteur 5, tout en gardant un code au plus clair et compréhensible.

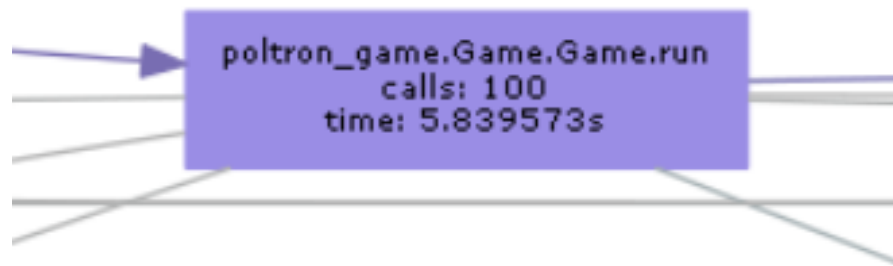
### 3.5. Remarques sur les résultats obtenus

**Éviter l'overhead des classes est providentiel** Au cours du développement du moteur et de ses optimisations, notre moteur initial utilisant des classes pour les moindres structures, comparé à une version de celui ci n'utilisant que des structures de bases, était 60 fois plus lent après compilation que le second.

L'ajout des annotations et de multiples micro-optimisations sur l'ensemble du moteur pour répondre aux besoins de notre projet nous on fait gagner un temps fou sur le calcul de l'heuristique, et par extension sur le calcul de chaque partie.



**Des gains de performances considérables** Sur notre moteur initial, sans compilation, une partie de base sans heuristique (M=5, N=5, C=2) durait 0.60s en moyenne.



Sur notre moteur actuel, une partie similaire, sans compilation, quasiment sans heuristique, dure désormais 0.05s en moyenne. Une exécution 12x plus rapide.

Cette meme partie atteint ensuite une durée moyenne de 0.0006s après compilation. Une execution 83x plus rapide par rapport au moteur actuel, et 1000x plus rapide par rapport au moteur initial de départ sans compilation.



## 4. Intelligence Artificielle - Heuristique

Ce chapitre est actuellement en cours d'écriture.

## 5. Analyse - Exploration

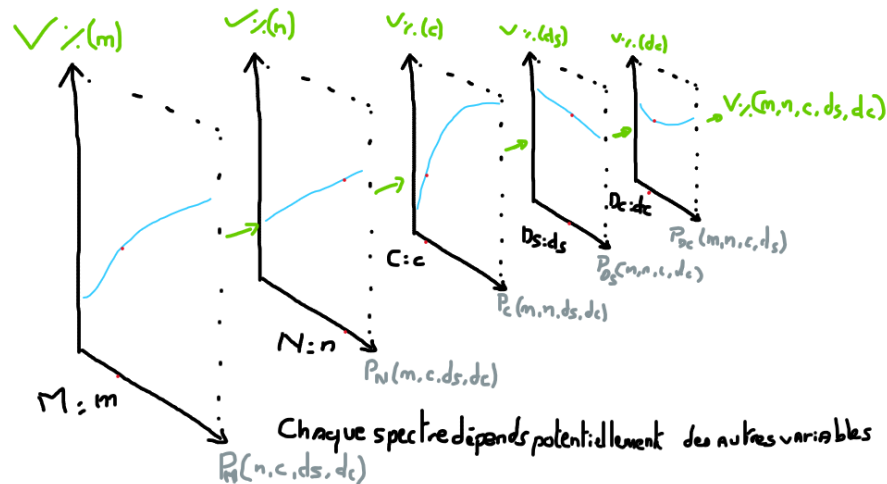
### 5.1. Nécessités

**Déterminer les facteurs d'une victoire** L'objectif final de ce projet est de déterminer les meilleurs paramètres initiaux permettant de maximiser le taux de victoires de la coalition.

Cela étant dit, notre objectif pour y parvenir est d'utiliser l'outil de l'analyse statistique, mais sur les données d'un demi-million de parties potentielles avec une demi douzaine de facteurs différents, par où commencer ?

### 5.2. Problème

**Comment déterminer les bonnes corrélations ?** Faire des statistiques à partir de l'intégralité de nos données permet de déterminer des tendances générales, mais sur notre cas où nous avons 5 dimensions de données indépendantes, et donc potentiellement des variations de ces corrélations à chaque modification infime de n'importe quel facteur, comment pourrions nous analyser relativement efficacement l'évolution de ces tendances pour tenter de déterminer de potentielles corrélations cachées entre plusieurs variables ?



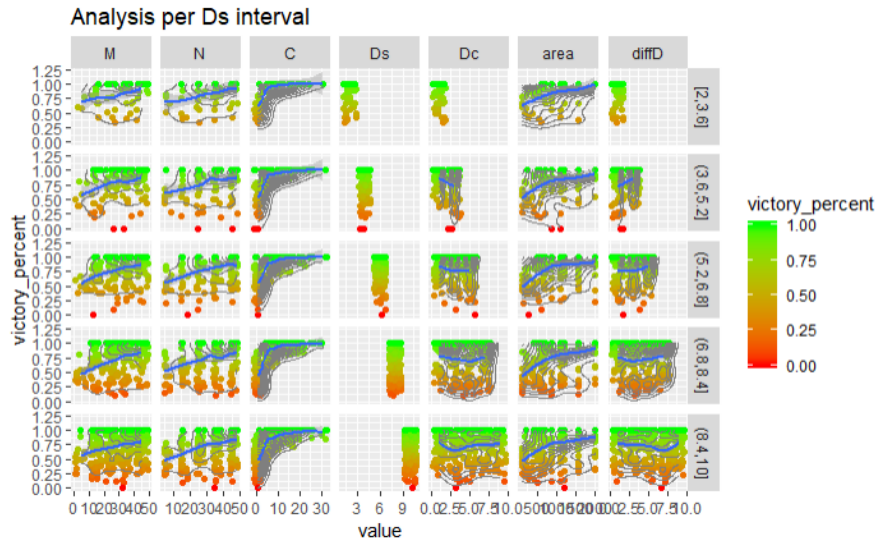
Comment pourrions-nous analyser nos données pour pouvoir y déceler des informations de la façon la plus efficace et complète possible sur autant d'axes ?

### 5.3. Approche utilisée

**Analyse par tranches** La quantité inconnue de données que nous avons pour chaque variable, construire une matrice à 5 dimensions serait prohibitif pour nos moyens actuels, autant en espace mémoire qu'en temps de calcul, d'analyse et de génération. C'est pour

cette raison que nous avons opté pour une simple analyse par intervalles de données présentes.

Analyser les tendances de victoire en fixant une variable ou plusieurs variables à la fois et en scindant nos données en 5 intervalles de tailles équivalentes nous permet d'analyser la progression des spectres entre de grandes variations des variables en questions et d'avoir une idée générale des relations entre variables.



## 5.4. Remarques sur les résultats obtenus

**Les données sont parlantes** Voir progresser les intervalles de données disponibles en fonction des intervalles de chaque variable et les voir se chevaucher petit à petit permet vraiment d'avoir une meilleure idée de ce que représente chaque intervalle dans la totalité des données présentes. De plus, la comparaison aisée entre les différents spectres à différents intervalles montrent bel et bien si les spectres changent beaucoup ou non selon tel ou tel intervalle d'une variable et permet de déterminer l'influence de cette variable sur ces spectres ou non.

## 5.5. Pistes d'amélioration

**Génération d'un profil 5D voire n-D de probabilités !** Comme dit plus haut, à partir de ce genre de données il paraîtrait tout naturel de tenter de modéliser un spectre 5D de probabilités permettant de déterminer automatiquement la probabilité de victoire d'un couple de paramètres initiaux arbitraires.

Cela pourrait d'ailleurs être un sujet bien chargé très intéressant à implémenter et à travailler qui pourrait permettre de s'intéresser à des sujets peu communs comme les tenseurs et l'interpolation !

**Cinquième partie**

**Analyse des données générées**

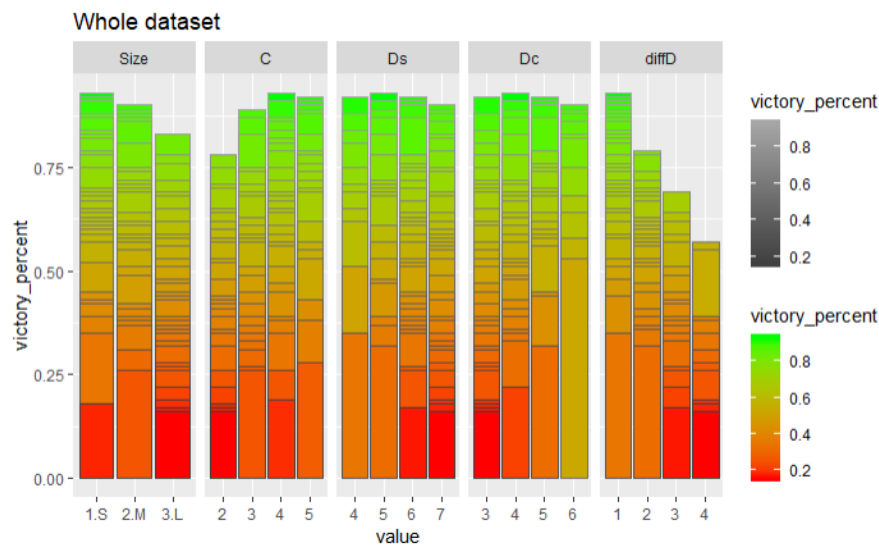
## 6. Analyse des données de l'IA

**Paramètres initiaux** Cette simulation as été réalisée avec 100 réitérations pour chaque combinaison possible avec les paramètres suivants.

```
Tested Ds values: {4, 5, 6, 7}
Tested Dc values: {3, 4, 5, 6}
Tested coalition sizes: {2, 3, 4, 5}
Smallest map dimensions: 8x8
Largest map dimensions: 22x22
Time estimated 0d 3h 33m 41s | 25.8% @ 1.44s/game curr (1.5,4,5,3) sim#3094
```

On remarquera que la profondeur de recherche as une petite fourchette dans notre espace de recherche pour des raisons de temps de calcul. Une analyse plus poussée dans de petites cartes pourrait être intéressante.

### 6.1. Analyse d'ensemble

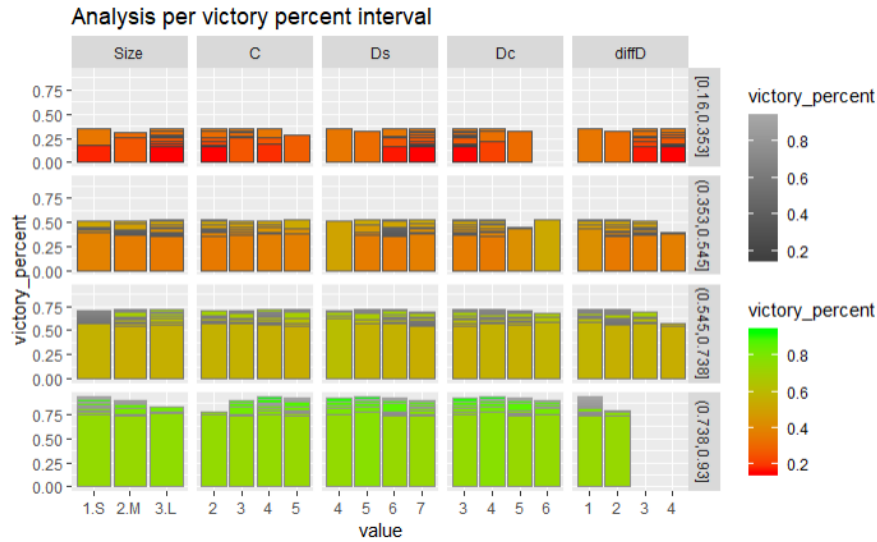


**Des données tendances variées** Nous pouvons d'abord remarquer la présence de cinq tendances générales de données :

- Plus la taille de la map est grande, plus les chances de victoires diminuent.
- Plus la taille de la coalition augmente, plus les chances de victoires semblent augmenter en général.
- Plus Ds augmente, et plus le chances minimales de victoire diminuent.
- Inversement pour Dc, celles-ci augmentent avec Dc.
- Plus la différence d'intelligence entre le joueur solo et la coalition augmente, plus les chances générales de victoires diminuent.

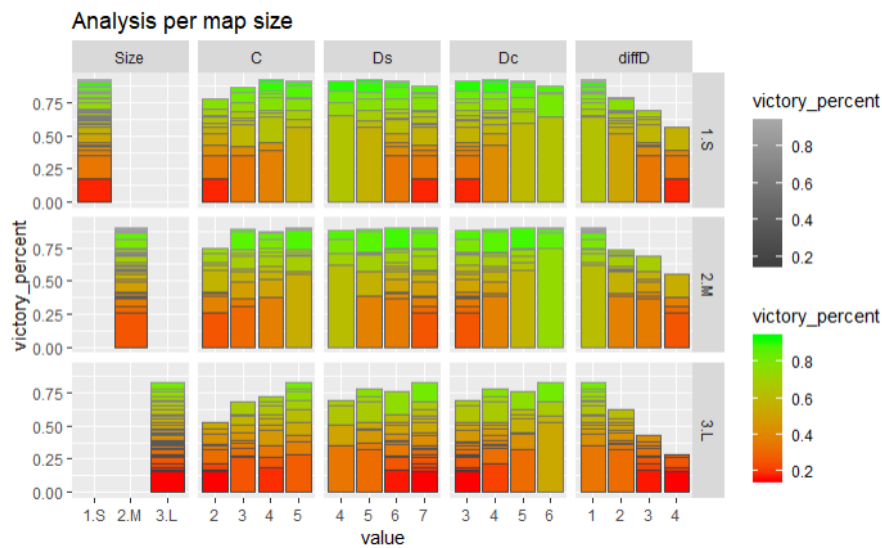
## 6.2. Analyses détaillées

### 6.2.1. Répartitions des pourcentages de victoires



**L'écart d'intelligence semble être primordial** Les victoires les plus assurées semblent être principalement réparties sur des couples de paramètres où la coalition et le joueur solo ont des intelligences proches, ainsi que lorsque Ds et Dc sont plus élevés. Les risques de défaites semblent être limités par de plus grandes valeurs de C et Dc, et de faibles valeurs de Ds. De plus, à haut pourcentage de victoires, la taille de la carte semble être un facteur important, où la coalition semble avoir l'avantage sur des cartes plus petites.

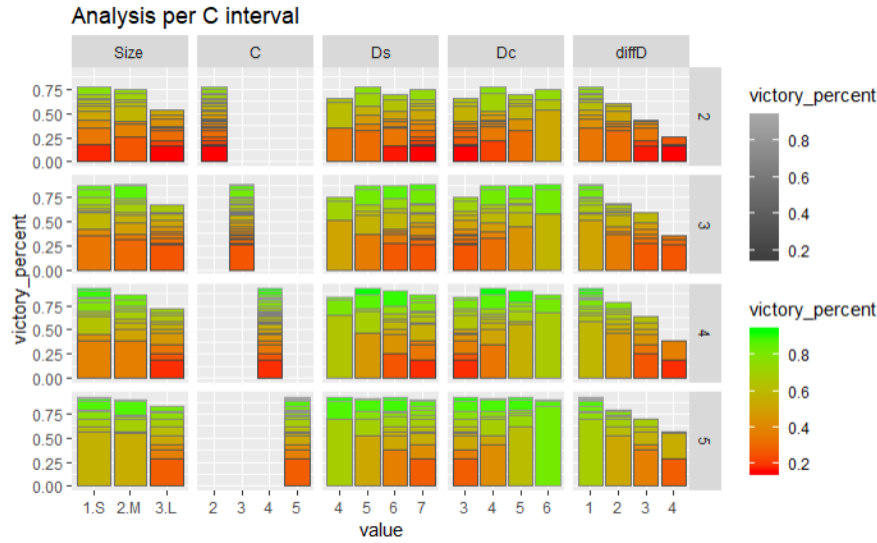
### 6.2.2. Découpage du spectre selon des intervalles de la taille de la carte



**Petites cartes avantageuses, grandes cartes handicapantes** Nous pouvons remarquer que sur les petites cartes, les chances maximales de victoires sont exacerbées sur tous les facteurs comparé aux autres tailles.

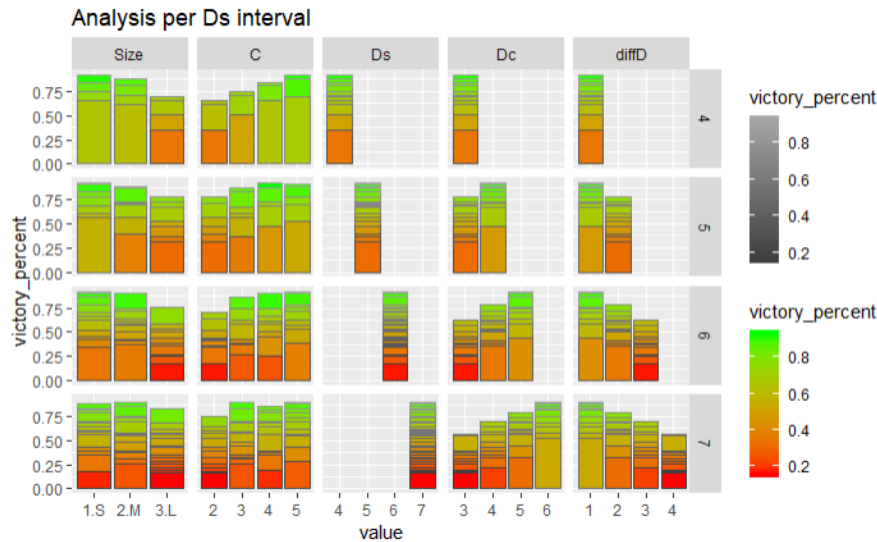
En revanche, les grandes cartes semblent avoir un effet négatif sur les chances minimales de victoire sur tous les facteurs.

### 6.2.3. Découpage du spectre selon des intervalles de C



**Le surnombre semble avantageux** Les chances de victoire générales semblent augmenter avec de plus grandes valeurs de C de façon homogène.

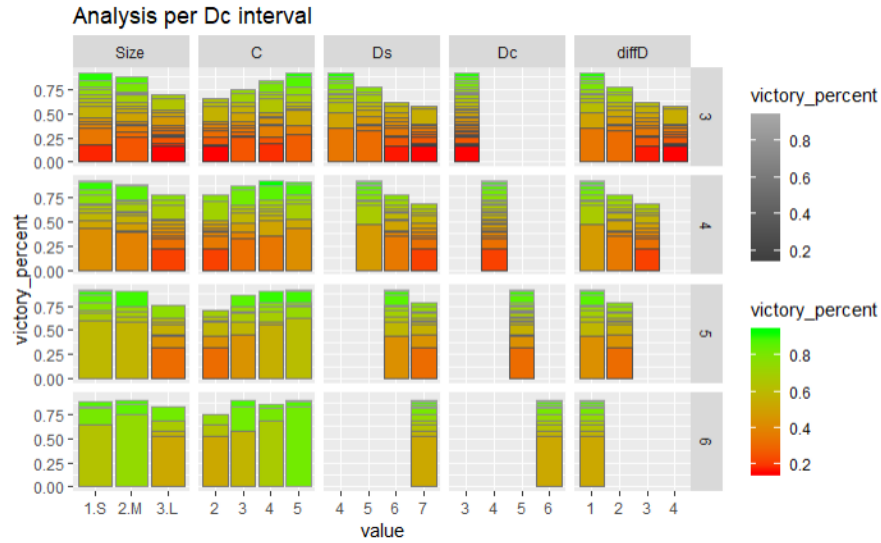
### 6.2.4. Découpage du spectre selon des intervalles de Ds



**Un joueur solo intelligent est plus difficile à battre** Les valeurs de Ds limitent beaucoup les données analysables par intervalle car le nombre de parties jouées avec de faibles valeurs de Ds est bien inférieur au nombre de parties jouées à de plus grandes valeurs de Ds. Ceci explique les nuances supplémentaires apparaissant à haute valeur de Ds.

Nous pouvons cependant remarquer que de plus grandes valeurs de Ds semblent diminuer les chances minimales de victoires.

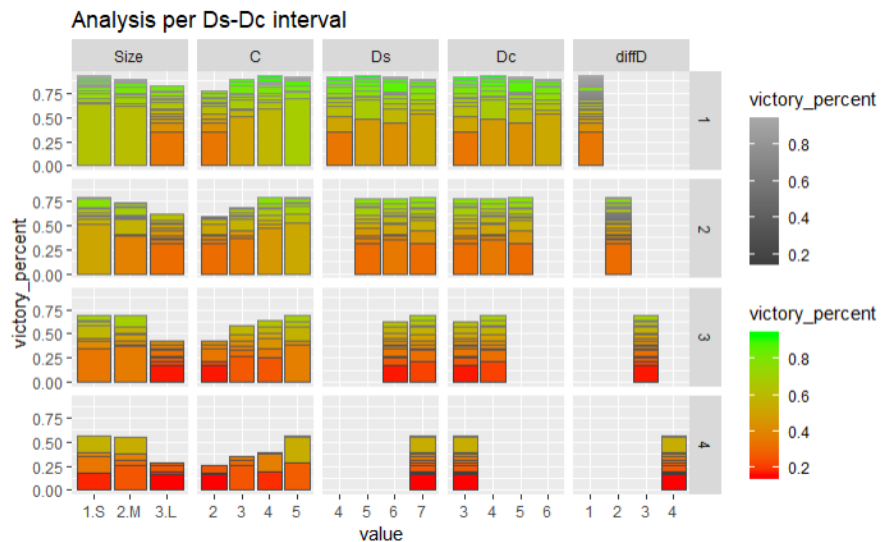
### 6.2.5. Découpage du spectre selon des intervalles de Dc



**Une coalition intelligente bats le joueur plus facilement** Inversement à Ds, il y a plus de parties jouées avec des valeurs faibles de Dc que hautes, ce qui explique les différences de nuances entre elles.

Cependant, Les plus grandes valeurs de Dc semblent maximiser les chances générales de victoire.

### 6.2.6. Découpage du spectre selon des intervalles de la différence de niveau



**L'écart d'intelligence est primordial** Nous pouvons clairement constater que l'écart d'intelligence diminue fortement les chances générales de victoires de façon drastique, passant de 30-90% pour  $diffD = 1$  à 20-55% pour  $diffD = 4$ .

Nous pouvons aussi constater une légère augmentation de ces chances de victoires quand Dc et Ds sont maximisées, ce qui pourrait indiquer que l'écart relatif ( $diffD = Ds/Dc$ ) d'intelligence pourrait être une mesure plus précise que l'écart absolu que nous utilisons ici ( $diffD = Ds - Dc$ ).



### 6.2.7. Conclusions d'analyse

**À niveau égal, rien ne va plus !** D'après nos résultats, nous pouvons constater que plus la différence relative d'intelligence est petite, plus la coalition a de chances d'écraser le joueur solo.

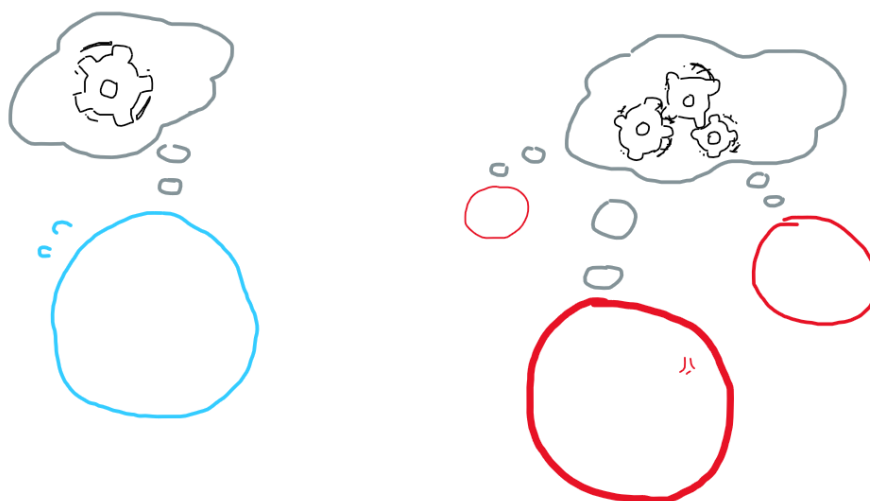
De plus, et ce surtout à partir d'un écart d'intelligence assez important, la taille de la carte semble indiquer que les cartes petites semblent favoriser la coalition, et que les grandes cartes semblent favoriser fortement le joueur solo.

En revanche, le surnombre semble être un avantage en toutes circonstances, mais son effet semble plafonner plus vite si l'écart d'intelligence est moindre.

La pire coalition possible serait une coalition qui joue quasiment au hasard, avec peu d'équipiers et sur une grande carte.

En revanche, la coalition idéale semble être une coalition en surnombre, d'intelligence égale au joueur solo, et sur une petite carte. Mais même juste à niveau égal, sur toute carte moyenne ou petite, la coalition est statistiquement meilleure.

C'est une bataille de cerveau.



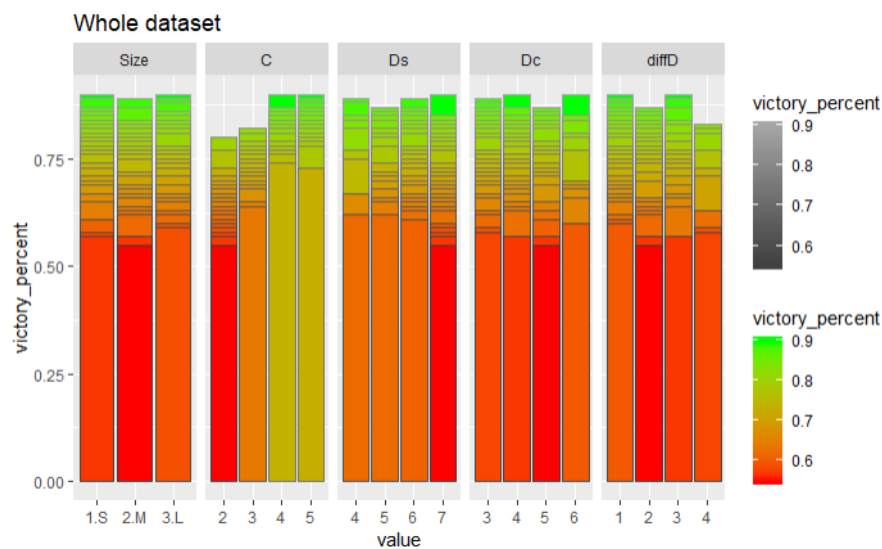
## 7. Analyse des données du modèle

**Paramètres initiaux** Cette simulation as été réalisée avec 100 réitérations pour chaque combinaison possible avec les paramètres suivants.

```
Tested Ds values: {4, 5, 6, 7}
Tested Dc values: {3, 4, 5, 6}
Tested coalition sizes: {2, 3, 4, 5}
Smallest map dimensions: 8x8
Largest map dimensions: 22x22

Time estimated 0d 0h 0m 0s | 100.0% @ 0.013s/game curr (1.S,5,5,3) sim#11999
```

### 7.1. Analyse d'ensemble

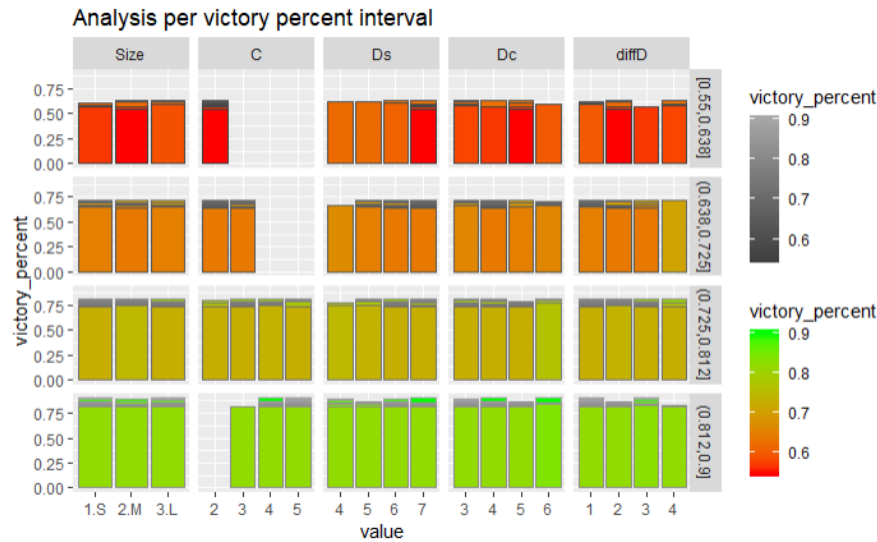


**Un seul facteur sort du lot** À commencer par le facteur C qui est le seul avec semble vraiment avoir de nette relation avec le pourcentage de victoires.

Les autres facteurs ne semblent pas particulièrement affecter les chances de victoires en général pour le moment.

## 7.2. Analyses détaillées

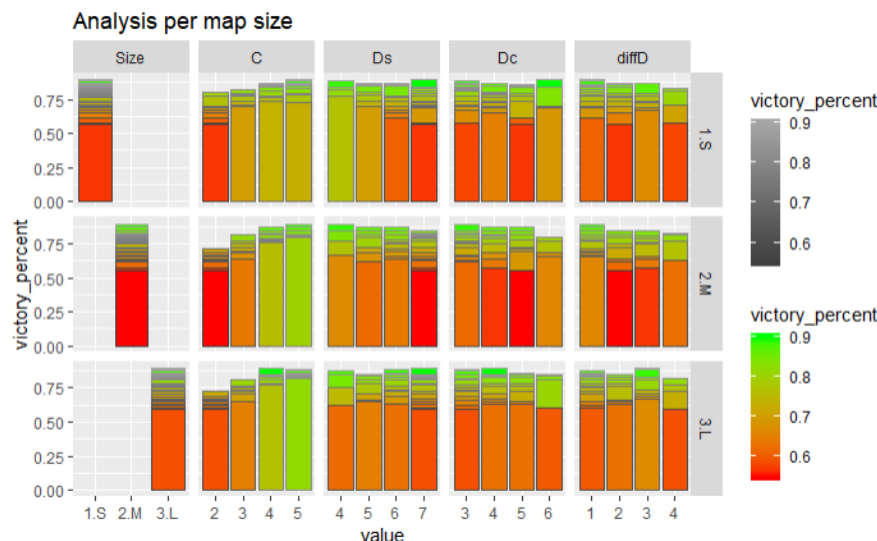
### 7.2.1. Répartitions des pourcentages de victoires



**Une évolution qui suit la distribution de C** Nous pouvons bel et bien constater plusieurs ensembles de données selon différentes valeurs de C.

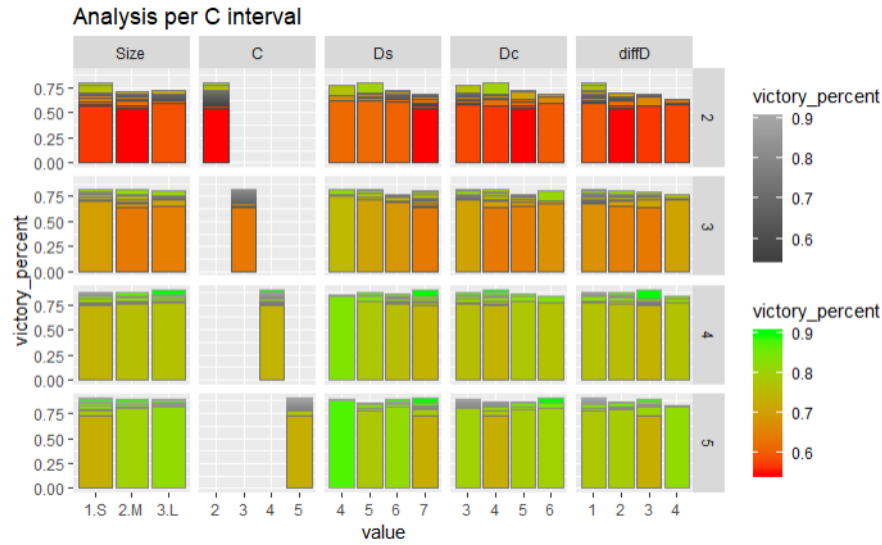
Les autres facteurs restent relativement homogènes, ceux ci ne semblent donc pas véritablement influencer la fin des parties.

### 7.2.2. Découpage du spectre selon la taille de la carte



**Aucun impact réel décelé** La taille de la carte ne semble pas influencer sur les pourcentages de victoire.

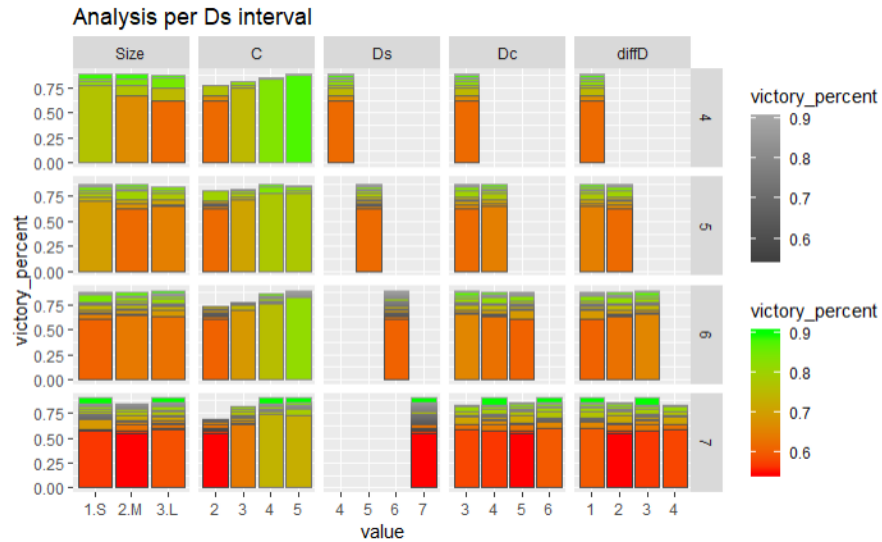
### 7.2.3. Découpage du spectre selon des intervalles de C



**C est définitivement un facteur MAJEUR** Cette représentation montre bien l'influence de C sur le pourcentage de victoire, aussitôt  $C \geq 3$  atteinte, les chances de victoires enregistrées sont au minimum de 60-80%.

Ceci dit, les chances de victoires semblent stagner vers 75-90% à des valeurs plus élevées. Ce qui reste tout de même majoritairement favoriser la coalition.

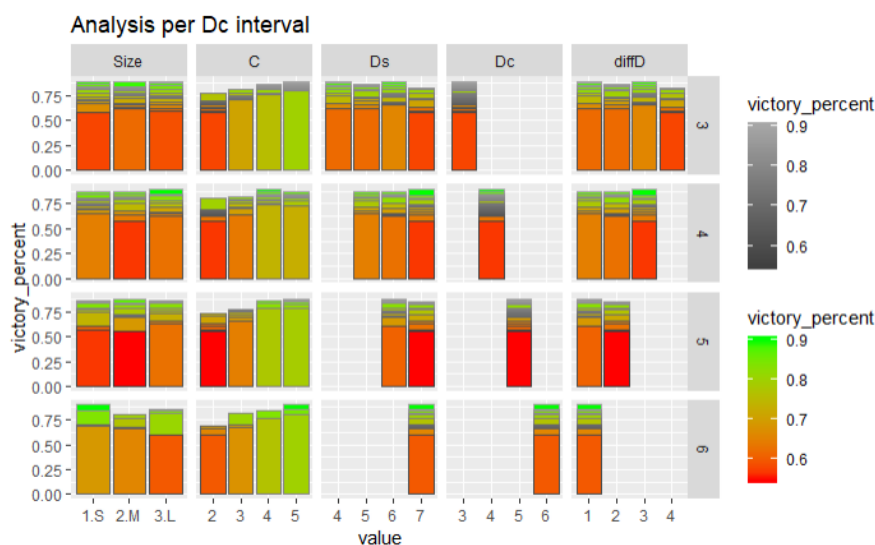
### 7.2.4. Découpage du spectre selon des intervalles de Ds



**Une légère diminution des chances de victoires** Ds ne semble vraiment pas influencer sur les différents spectres non plus, cependant, l'écart de probabilité se creuse avec des valeurs plus grande de Ds, et semble accentuer les tendances de C.

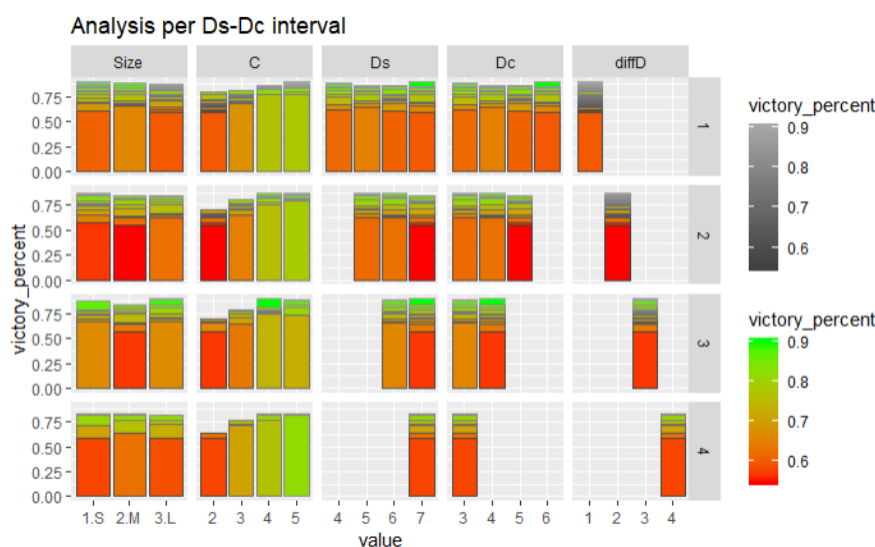
À plus grandes valeurs de Ds, les probabilités de victoires semblent plus nuancées et défavoriser la coalition.

## 7.2.5. Découpage du spectre selon des intervalles de Dc



**Aucun impact réel décelé** Dc ne semble vraiment pas influencer sur les différents spectres, par conséquent, Dc ne semble pas avoir d'impact tout court sur notre distribution de probabilité.

## 7.2.6. Découpage du spectre selon des intervalles de la différence de niveau



**Toujours aucun impact réel décelé** La différence d'intelligence ne semble pas non plus influencer particulièrement sur les chances de victoires.

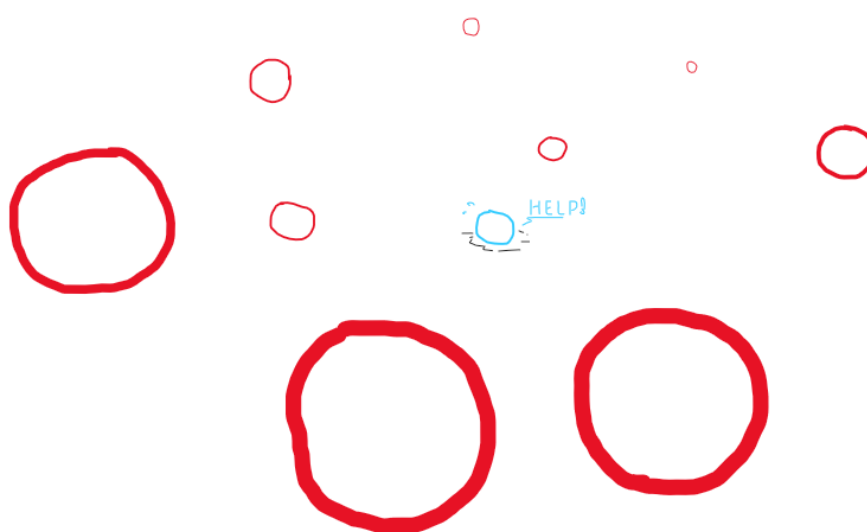
La quantité de nuances est explicable par la plus grande quantité de parties jouées à différence de 1 qu'à grande différences, car Dc doit toujours être inférieur à Ds et par conséquent, sur de faibles valeurs de Ds ( $Ds \in [2; 4]$ ), il n'est pas possible d'avoir d'écarts de 4 avec  $Dc > 0$ .

### 7.2.7. Conclusions d'analyse

**Le contrôle de la carte est le facteur le plus important d'après notre modèle** La seule véritable corrélation que nous ayons pu déceler ici entre variables et pourcentages de victoire est en fonction du nombre de joueurs présents sur la carte.

Si nous y réfléchissons bien, cela fait même sens !

Avec une stratégie purement défensive (d'esquive de murs), et un unique joueur dans son équipe, un joueur seul a plus de chance d'éliminer son équipe sur la durée qu'une équipe nombreuse jouant aussi en pure défensive et contrôlant plus de surface au total.



Ce modèle ne colle malheureusement pas au comportement complet de notre intelligence artificielle, mais permet d'approximer à moindres coûts une stratégie de survie pure, comme lorsqu'un joueur est bloqué dans une zone sans adversaires.

Une optimisation en mélangeant ces deux modèles pourrait être utile pour accélérer les calculs.

**Sixième partie**

**Problèmes, tests et  
expérimentations**

## 8. Problèmes rencontrés

Ce chapitre est actuellement en cours d'écriture.



## 9. Conclusion

Ce chapitre est actuellement en cours d'écriture.

**Résumé des objectifs au résultat final**

**Résumé sur l'enrichissement personnel**

**Résumé perspectives envisagées, appréciation perso, poursuite..**

## **Septième partie**

### **Annexes**

# 10. Analyse - Simulation

## 10.1. Nécessités

**Tenter de modéliser un système à priori intelligent** Les calculs d'heuristique sont chers, et il est parfois utile pour optimiser la prédiction ou la détection d'anomalies de tenter de trouver un modèle mathématique qui décrit bien l'évolution de notre système.

Mais si les calculs d'heures sont chers, ils sont malheureusement au premiers abord nécessaires, pour pouvoir observer des phénomènes dans leur globalité avant de tout analyser. Pour tenter de trouver un modèle il va donc falloir beaucoup de données pour trouver de potentielles corrélations récurrentes.

## 10.2. Problème

**À quel point telles données sont elles pertinentes ?** Afin de pouvoir affiner notre modèle et le faire ressembler un maximum au comportement de notre véritable phénomène, il est important de déterminer l'importance de chaque facteur dans son comportement.

Comment déterminer par la suite l'équivalence de cette importance dans notre autre visualisation du problème ? Sont-elles comparables ?

**Quels types de modèles peuvent nous créer ?** Il existe une infinité de fonctions possibles, correspondant à notre profil recherché sur notre intervalle de données. Si notre modèle doit être capable d'extrapoler, l'intuition et les analogies au monde physique peuvent-ils être suffisants ?

La réponse à cette question est évidemment complexe. Mais aussi évidente : On ne peut pas prédire avec exactitude quelque chose sur laquelle nous n'avons aucune donnée. Par conséquent, il va falloir partir du principe que les données suivantes ressembleront à une tendance connue qui correspond déjà aux données présentes.

Mais comment déterminer laquelle ?

Comment pourrions-nous déterminer un type de modèle et ses paramètres pour représenter le même comportement que notre phénomène connu ?

## 10.3. Approches possibles

**Mathématiques pures** Avec des mathématiques pures, et à partir de suffisamment de données d'entrées, nous devrions pouvoir construire un spectre de probabilités de victoire en fonction des paramètres initiaux.

Cela devrait permettre une prédiction des plus fidèles de notre comportement sur l'intervalle connu, mais l'extrapolation sur un spectre de probabilités nous semble être d'un très haut niveau de mathématique que nous n'avons malheureusement pas dans notre équipe.

La prédiction en données connues serait donc instantanée mais l'extrapolation impossible.

**Modèle simulé inspiré par la physique** Avec de la physique il est possible de tenter de raisonner différemment, et de calculer potentiellement plus rapidement le même genre de résultats que le calcul complet de notre phénomène de départ.

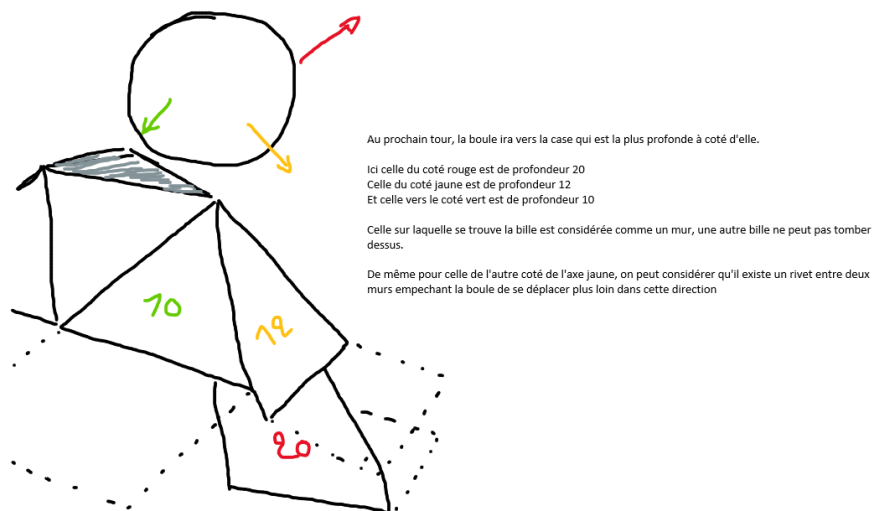
De plus, cela permettrait aussi potentiellement de mettre en équation le comportement des acteurs de notre phénomène, ici les IA, et de potentiellement trouver un modèle simplifié et fonctionnel pour leur heuristique.

Ici les équations pourraient permettre de l'extrapolation relativement aisément, mais la précision de notre modèle va nécessiter un lourd travail de réglages pour trouver l'équilibre entre l'influence des paramètres dans le modèle des IA et celle sur les paramètres dans le modèle simulé.

Un exemple parlant est la traduction de l'influence de la profondeur de recherche des IA vers le modèle physique, où il n'y aura pas d'IA.

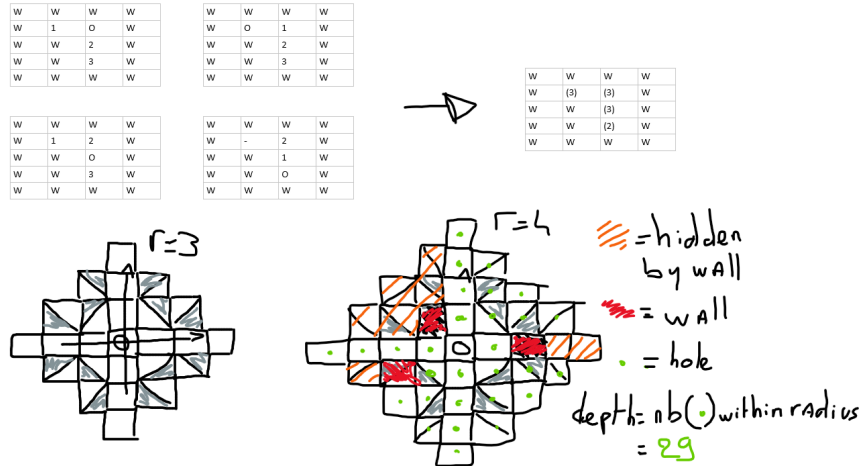
## 10.4. Approche utilisée

**Modèle physique** Le temps et l'expérience limitée des membres du groupe dans le domaine de la simulation nous ont forcés à partir pour le modèle inspiré de la physique.



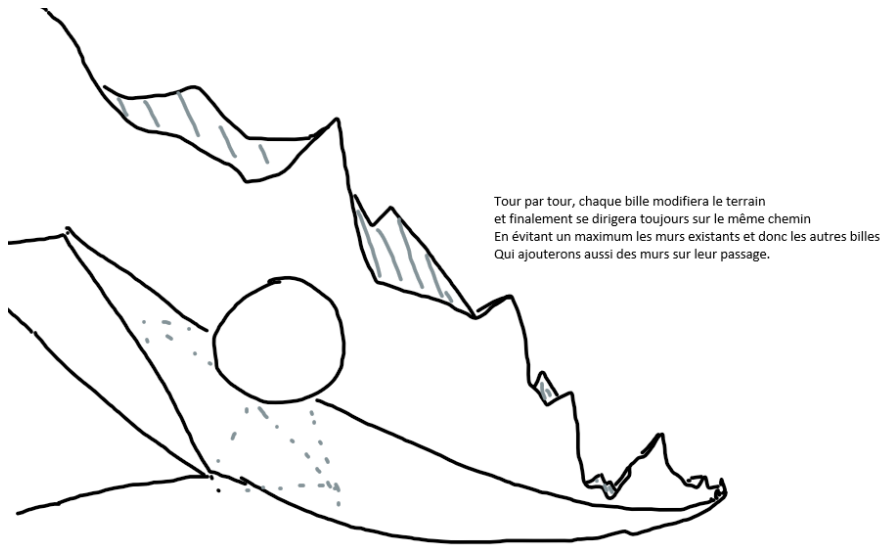
Nous avons donc pris les joueurs pour des billes (littéralement), et les considérons désormais en roulis perpétuel vers la pente la plus accentuée qui s'offre à eux.

À la suite de leur roulis (changement de case), un mur se crée à leur position actuelle, les forçant à continuer de rouler au tour suivant.



Ces murs sont de matériau friable, comme du sable, et une fois posés, remplissent les trous environnants d'une légère quantité de sable sans jamais les boucher, réduisant ainsi leur pente.

Une bille ne peut plus rouler si elle est entourée de murs, autrement dit, si elle n'a plus de pente sur laquelle rouler. Elle est alors retirée du jeu et considérée hors jeu.



Ce système devrait favoriser le contrôle de la carte de façon naturelle, car les billes seront donc naturellement inclinées à se diriger vers les endroits les plus profonds, c.à.d. ceux ayant le plus d'espace libre, et continueront toujours de rouler tant qu'elles n'auront pas atteint de cul de sac.

Pour tenter de donner une dimension d'équipe, nous avons attribués une légère force attirant la coalition vers la position du joueur pour départager deux cases de même profondeur.

Un barycentre de force pourrait être nécessaire pour calculer la force inverse pour le joueur seul, et cela nous a semblé potentiellement trop coûteux en temps de calculs supplémentaires pour rendre le modèle potentiellement viable. L'idée reste à tester.

Ce modèle est basé sur l'heuristique de base du projet, qui est la maximisation de la surface contrôlée par les équipes à chaque tour. L'IA a évolué depuis.

## 10.5. Remarques sur les résultats obtenus

**Des résultats qui ne collent pas à notre modèle IA** Notre physique semble être trop indépendante de l'intelligence des joueurs, les données de l'IA nous montrent une différence plus portée sur l'écart d'intelligence que sur le reste.

Ceci est sûrement explicable par le fait que ce modèle joue sur un principe purement défensif tandis que notre IA alterne parfois entre défense et agression.

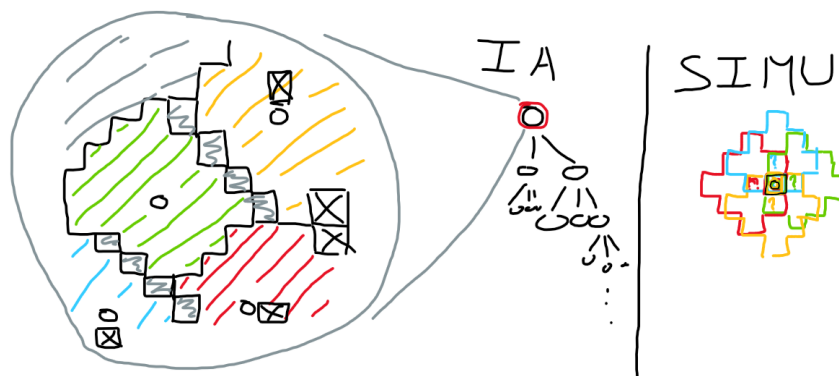
Il serait peut être possible cependant d'utiliser notre modèle en optimisation pour les IA en totale autarcie et économiser en temps de calcul quand aucune stratégie offensive n'est désormais utile.

**Un semblant de stratégie !** Nous avons pu constater que malgré l'absence de prédiction de la part des billes, celles-ci semblaient avoir parfois des semblants de stratégies, par exemple, nous avons pu surprendre le joueur solo à coincer un adversaire dans un couloir de deux cases puis lui faire une queue de poisson en sortie !

Même sans prédiction, nous arrivons donc à recalculer des mouvements potentiellement stratégiques, il y a donc espoir de pouvoir affiner notre modèle plus encore.

**Une optimisation de temps de calcul efficace !** Là où notre IA est forcée de calculer la zone de contrôle d'un joueur jusque  $3^C * D_s$  fois dans le pire des cas (tous les joueurs peuvent aller dans 3 directions à chaque tour, sur  $D_s$  tours), sur l'intégralité de la carte importante pour chaque joueur simulé à chaque simulation, notre système se contente d'un rayon de recherche de profondeur qui calcule en une fois l'intérêt d'une case et de ses alentours, et ce sur les quatre cases adjacentes à la bille.

Le calcul de la zone ne néglige les cases inutiles et donc s'arrête soit en cas de rayon atteint, soit en cas d'obstacle atteint dans sa propagation de comptage, ce calcul peut donc être de complexité  $O(1)$  en cas d'encerclement et sa moyenne baisse à mesure que le nombre de murs augmente.



À contrario, la recherche pour notre IA nécessite de calculer et propager autant de zones que de joueurs, incluant donc une plus grande surface à propager pour le calcul des zones d'un seul joueur, lors d'une unique étape de simulation.

## 10.6. Ordre de grandeur de la différence de vitesse de calcul

M	N	C	Ds	Dc
50	50	36	10	9

FIGURE 10.1. – Paramètres initiaux

**Des paramètres initiaux extrêmes** Cette partie est la partie la plus extrême de notre échantillon de données sortant de notre simulation modélisée, en sachant que celles ci ont toutes été répétées 100 fois pour avoir une certaine précision. Nous allons ici tenter d'estimer à grand renfort d'approximations la quantité de cases touchées par nos calculs pour la décision de nos joueurs afin de pouvoir comparer la différence d'efficacité entre nos deux modèles.

**Les constantes** Nous déterminons d'abord les constantes qui nous serviront à simplifier nos équations :

- $D$  représente la profondeur moyenne de recherche des joueurs
- $t_{max}$  représente le nombre maximum de tours pouvant être joués si personne ne meurt tout le long de la partie.

$$D = \frac{C * Dc + 1 * Ds}{C + 1} \quad (10.1)$$

$$\approx 9 \quad (10.2)$$

$$t_{max} = \frac{M * N}{C + 1} \quad (10.3)$$

$$\approx 68 \quad (10.4)$$

### 10.6.1. Modèle IA

$$nbCases(t, M, N, C) = M * N - (C + 1) * t \quad (10.5)$$

$$nbCases(t) = 2500 - 37 * t \quad (10.6)$$

$$nbCases/tour/joueur(t, D) = \int_t^{t+D} nbCases(x) dx \quad (10.7)$$

$$= [2500 * t - 37 * \frac{t^2}{2}]_t^{t+D} \quad (10.8)$$

$$= 2500(t + D - t) - 37 * \frac{(t + D - t)^2}{2} \quad (10.9)$$

$$nbCases/tour/joueur(D) = 2500 * D - \frac{37}{2} * D^2 \quad (10.10)$$

$$totalCases(t_{max}, C, D) = (C + 1) * t_{max} * nbCases/tour/joueur(D) \quad (10.11)$$

$$= (C + 1) * t_{max} * (2500 * D - \frac{37}{2} * D^2) \quad (10.12)$$

$$= 37 * 68 * (2500 * 9 - \frac{37}{2} * 9^2) \quad (10.13)$$

$$= 2516 * (22500 - 18.5 * 9^2) \quad (10.14)$$

$$= 2516 * (22500 - 1498.5) \quad (10.15)$$

$$= 2516 * 21001.5 \quad (10.16)$$

$$totalCases(t_{max}, C, D) \approx 52839774 \quad (10.17)$$



### 10.6.2. Modèle simulé

$$nbCasesInRadius(D) = \int_0^D 4 * x dx \quad (10.18)$$

$$= [4 \frac{x^2}{2}]_0^D \quad (10.19)$$

$$nbCasesInRadius(D) = 2D^2 \quad (10.20)$$

$$totalNbMurs(t, C) = t * (C + 1) \quad (10.21)$$

$$ratioMur/case(t, M, N, C) = \frac{totalNbMurs}{M * N} \quad (10.22)$$

$$ratioMur/case(t, M, N, C) = \frac{t * (C + 1)}{M * N} \quad (10.23)$$

$$nbMurInRadius(t, M, N, C, D) = ratioMur/case(t, M, N, C) * nbCasesInRadius(D) \quad (10.24)$$

$$nbMurInRadius(t, M, N, C, D) = \frac{t * (C + 1)}{M * N} * 2D^2 \quad (10.25)$$

$$casesLibreInRadius(t, M, N, C, D) = nbCasesInRadius(D) - nbMurInRadius(t, M, N, C, D) \quad (10.26)$$

$$casesLibreInRadius(t, M, N, C, D) = 2D^2 - \frac{t * (C + 1)}{M * N} * 2D^2 \quad (10.27)$$

$$casesLibreInRadius(t, M, N, C, D) = 2D^2 * (1 - \frac{t * (C + 1)}{M * N}) \quad (10.28)$$

$$nbCases/player/turn(t, M, N, C, D) = 3 * casesLibreInRadius(t, M, N, C, D) \quad (10.29)$$

$$= 3 * 2D^2 * (1 - \frac{t * (C + 1)}{M * N}) \quad (10.30)$$

$$nbCases/player/turn(t, M, N, C, D) = 6D^2 * (1 - \frac{t * (C + 1)}{M * N}) \quad (10.31)$$

$$nbCases/turn(t, M, N, C, D) = (C + 1) * nbCases/player/turn(t, M, N, C, D) \quad (10.32)$$

$$nbCases/turn(t, M, N, C, D) = (C + 1) * 6D^2 * (1 - \frac{t * (C + 1)}{M * N}) \quad (10.33)$$

$$totalCases(t_{max}, M, N, C, D) = \int_0^{t_{max}} nbCases/turn(t, M, N, C, D) dt \quad (10.34)$$

$$= (C + 1) * 6D^2 * \int_0^{t_{max}} 1 - \frac{t * (C + 1)}{M * N} dt \quad (10.35)$$

$$= (C + 1) * 6D^2 * [t - \frac{(C + 1)}{M * N} * \frac{t^2}{2}]_0^{t_{max}} \quad (10.36)$$

$$= (C + 1) * 6D^2 * (t_{max} - \frac{(C + 1) * t_{max}^2}{2 * M * N}) \quad (10.37)$$

$$= 37 * 486 * (68 - \frac{37 * 68^2}{5000}) \quad (10.38)$$

$$= 17982 * (68 - 34) \quad (10.39)$$

$$totalCases(t_{max}, M, N, C, D) = 611388 \quad (10.40)$$

### 10.6.3. Comparaison des deux modèles

**Nous pouvons calculer notre speedup** Grâce aux résultats ci dessus, nous pouvons calculer le speed up entre nos deux modèles.

$$\frac{totalCases_{IA}}{totalCases_{simu}} = \frac{52839774}{611388} \quad (10.41)$$

$$\frac{totalCases_{IA}}{totalCases_{simu}} \approx 86 \quad (10.42)$$

Pour ce cas extrême, notre simulation teste 86 fois moins de cases que notre IA de base. C'est énorme.

## 10.7. Pistes d'amélioration

**Un meilleur réglage des facteurs du modèle** Pour le moment nous avons considéré les facteurs de notre modèle relativement linéairement à partir des paramètres initiaux mais l'IA peut potentiellement réagir à d'autres facteurs que nous pourrions peut etre quantifier pour améliorer notre modèle.