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

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

Table des figures

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

Modèle de jeu Le moteur de jeu devra être le plus optimisé possible pour éviter de couter en temps de simulation pour l'heuristique et permettre le calcul d'un maximum de parties.

IA et son heuristique L'intelligence artificielle va devoir être capable de se défendre et d'attaquer l'équipe adverse de façon efficace et l'heuristique devra être optimisée au possible.

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	Fléxibilité	Contribuant 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;~~ ~~Une fois une tache réalisée celle-ci doit être rayé par un code couleur (rouge : fait mais à optimiser/ou pas sûr (souvent le cas)/vert : finis parfaitement;~~ ~~Une fois tous les éléments de la case fléxibilité 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"); Extremité 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 paranoide; ~~Heuristique~~ : ~~vérification de la zone de contrôle d'un camps sur le plateau~~; 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 interessantes 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

Paramètre de simulation	Signification
M	Longueur du terrain
N	Largeur du terrain
C	Nombre de joueurs dans la coalition
Ds	Intelligence du joueur solo
Dc	Intelligence des joueurs de la coalition

FIGURE 1.1. – Nomenclature des paramètres de jeu

1.3.2. Contraintes techniques

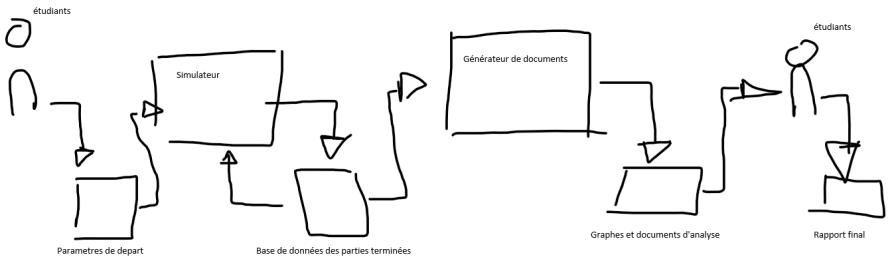
Temps imparti réduit Suite à une annulation de la matière puis à la réouverture de celle ci, le temps imparti pour ce projet as été considérablement amoindri.

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. Architecture



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 configuer 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 evalutation 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 + anayse 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	Completer 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 differentes doc sur le jeu, les differentes IA, exemple de code, amelioration possible, infos importante au "n" moment pour avoir un graphe interessant, etc.	3H	N/A	N/A
01/02/2019	Walid	Analyse du projet	Analyse des doccuments 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 prete avant de pouvoir determiner le format du cache
02/02/2019	Christopher	Création d'un prototype d'analyseur	Générateur de données aléatoire et rendu des graphes fonctionnels	6H	Avoir de vraies données pour affiner les affichages les plus interressants + implementer des graphes de comparaison de profondeur de recherche	N/A
02/02/2019	Vincent	Convertion 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 methodologie 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 toute les fonctionnalités et de l'architecture du jeu Tron*	10H	deplacements	N/A
03/02/2019	Christopher	Couche de gestion de la simulation	Génére 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 utilisé, préparé des 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 a 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 deplacements, 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 pensé du code du jeu, optimisation de celui-ci	Code optimisé à fond, 60 fois plus rapide qu'avant	4H	FIIIINIIIIIEEEE	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 facon differente	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'éxecution 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 details 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 les case les plus proches aux équipes. Ajout d'une méthode permettant la vérification des portées de chaques joueurs.	9H	Faire des tests avec un 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 meme partie extrême entre le modèle simulé et le modèle d'IA	2H	N/A	N/A
10/02/2019	Vincent	Algorithme Paranoide	Création de 2 versions d'algorithme paranoid; Optimisation de l'algorithme paranoide	13H	Optimiser le code; Changer les noms de variables/fonctions (nettoyer le code)	Beaucoup de partie ne peuvent pas être joué, 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 prediction (profondeur plus grande que son adversaire)	5H	N/A	L'ia se retrouve à jouer un coup letal, du à l'utilisation de l'algorithme minmax
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 prévus, ou trouver une heuristique compatible avec l'algorithme	L'heuristique renvois la taille zone, suivant qui l'execute, 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 systeme 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 générés 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 chacuns 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 délaisser 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 :
— IDLE pour coder en python

Vincent :
— Vim pour coder en python

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 IA

— random pour jouer des coups aléatoire lors de situations spécifiques.

1.6.2. Module simulation

— cython pour compiler et accélerer le temps d'éxécution
— PyCallGraph pour une représentation graphique du graphe d'appels, permettant de mieux localiser les endroits à optimiser dans notre programme.
— time pour estimer le temps restant avant completion des simulations
— sqlite3 pour l'interfacage avec la bdd sqlite

1.6.3. 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

Quatrième partie

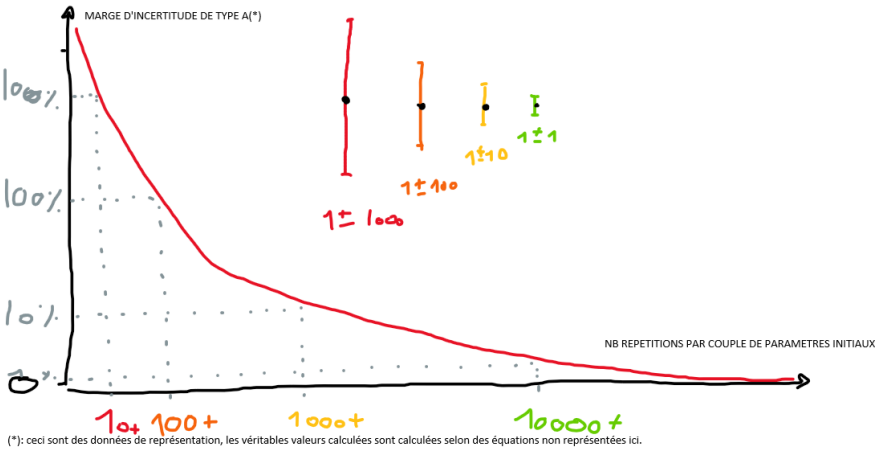
Réalisation

2. Simulateur

2.1. Nécessités

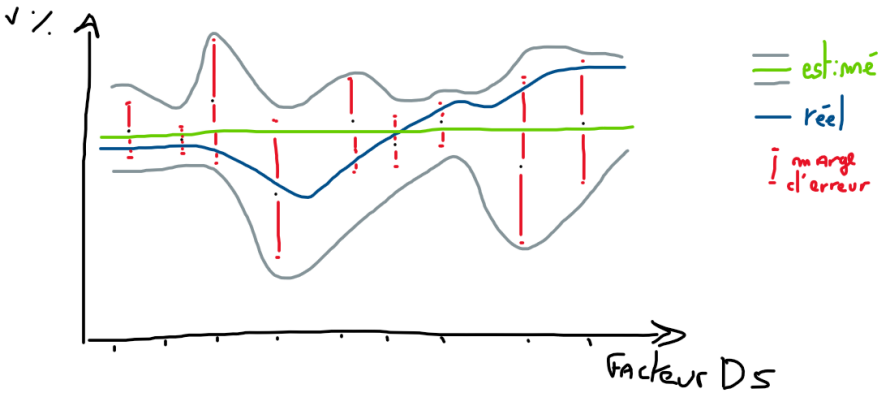
Tester de façon uniforme notre espace de recherche Afin de pouvoir avoir des statistiques les moins biaisées possible, il est nécessaire d'uniformiser nos simulations sur notre espace de recherche afin d'éviter la sous-représentation de certains couples de paramètres initiaux.

2.2. Problème



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

Cependant, quelques tentatives ne serait 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 faudrais donc répeter 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 graphe représente ci-dessus un exemple de mauvaise représentation potentielle, dûe à la différence de marge d'erreur.

Des données avec les memes 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 pourrais avoir considéré le point haut en incertitude du creux comme une anomalie comparés 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 initaux 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 dûes 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 necessaire 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 répeter 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'iterations 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'execution 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 génerer 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 sytèmes de calculs, mais notre Cython a malheureusement souffert de l'overhead Python de la librairie multiprocessing et l'éxecution s'est révélée plus lente que sans.

Une implémentation du multiprocessing directement en C ou via une libraire 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

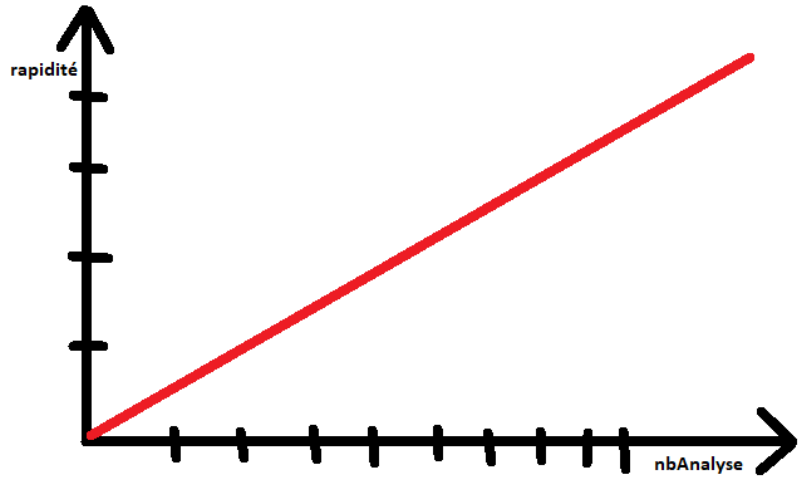
Un moteur de jeu efficace Dans le cadre de notre projet tutoré nous devons répondre à la problématique : "Combien faut-il d'idiots pour prendre l'avantage sur un joueur plus intelligent ?". Pour y répondre, nous avons besoin d'analyser beaucoup de parties différentes.

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 annoatations 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 incombes 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 modulable, 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'éxecution de leur code par cython. De ce fait, utiliser pleinement les classes python peut ralentir l'éxecution 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 éxecution 5 fois plus rapide en moyenne sur des parties de paramètres (M=10, N=10, C=4, Ds=4, Dc=3).

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

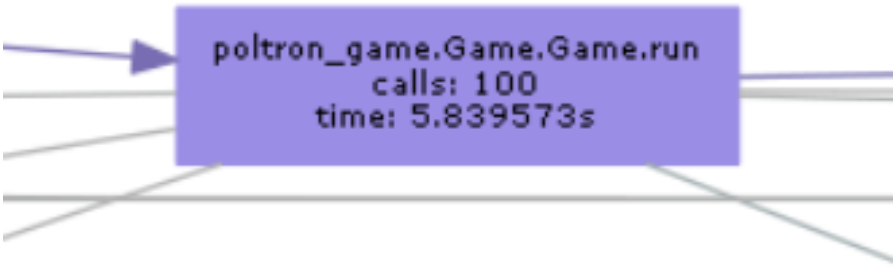
> Minimiser l'usage de classes à permis d'accélerer l'éxécution moyenne de notre moteur d'un facteur 5, tout en gardant un code au plus clair et compréhensible.

3.5. Remarques sur les résultats obtenus

Éviter l'overhead des classes est providentiel Au cours du dévelopement 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 éxecution 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 - Algorithme

4.1. Éléments techniques

Algorithme (4.1.2.) : Un algorithme est une suite finie et non ambiguë d'opérations ou d'instructions permettant de résoudre une classe de problèmes.
https ://fr.wikipedia.org/wiki/Algorithme

Arbre (4.1.2.) : Un arbre en informatique est consititué d'arêtes (appelé branche), d'états non finis (appelé noeud) et finis (appelé feuille). Chaques branches mênent à un état à l'aide d'une action à partir d'un état précédent (père). On appelle feuille un état qui ne possède pas d'état suivant (fils). La racine de l'arbre est appelé l'état initial.

Algorithme MinMax (4.1.2.) : L'algorithme minimax (aussi appelé algorithme MinMax) est un algorithme qui s'applique à la théorie des jeux1 pour les jeux à deux joueurs à somme nulle (et à information complète) consistant à minimiser la perte maximum (c'est-à-dire dans le pire des cas).
https ://fr.wikipedia.org/wiki/Algorithme_minimax

Profondeur de recherche (4.1.4.) : La profondeur de recherche dans les algorithmes s'appliquant à la théorie des jeux, est la limite du niveau qui peut être parcouru depuis l'état initial aux noeuds de l'arbre.

4.2. Nécessités

Comment gagner au jeu de Tron ? Pour jouer une partie, le programme doit savoir prendre une décision sur la direction que doit emprunter un joueur. La façon la plus pratique est de simuler tous les déplacements possibles à partir d'une position donnée, puis répéter ce processus sur plusieurs tours de jeu. Ce qui permet d'arriver à une partie finie ou un état du plateau qui pourrait exister.

4.3. Problème

Comment générer un arbre* d'états, dans une partie à plusieurs joueurs ? Il existe plusieurs styles d'algorithmes* permettant de résoudre des problèmes différents. Pour notre sujet, nous devons utiliser des algorithmes s'appliquant à la théorie des jeux. Le principe de l'algorithme MinMax* permettrait de trouver un déplacement idéal sur le plateau du jeu Tron, cet algorithme fonctionne pour les jeux à deux joueurs à sommes nulles, nous ne pouvons pas l'utiliser. En essayant de nous inspirer de son principe, nous pourrons générer un parcours dans l'arbre étant évalué par une heuristique*.

4.4. Approches possibles

Algorithme Paranoide L'algorithme paranoïde fonctionne sur le principe de celui de MinMax, une des différences qui existe entre ces deux algorithmes est qu'il s'applique sur la théorie des jeux pour plusieurs joueurs. Cet algorithme part du principe que tous les joueurs du terrain sont contre lui, tous les joueurs adverses choisiront donc la perte maximum du joueur et lui minimisera sa perte.

L'algorithme paranoïde à hauteur de sa profondeur de recherche* ne pourra pas se tromper. En revanche, il n'est pas garanti que le meilleur coup actuel ne soit pas le pire coup au tour suivant.

Algorithme de Monte-Carlo L'algorithme de Monte-Carlo fonctionne sur le principe de l'aléatoire. Cet algorithme n'as pas besoin d'une fonction d'heuristique explicite pour fonctionner, pour déterminer la valeur d'une situation, il lui suffit de determiner des coups au hasard, puis de jouer les états de parties résultantes de ces coups pour faire une moyenne des défaites et victoires. Les coups ayant une succession de coups le plus victorieux sont alors considérés comme les coups les plus intéressants potentiellement.

L'algorithme Monte-Carlo est très rapide, et donné un temps infini de calcul, converge vers le résultat d'un min max, mais sa version de base converge lentement et son approche algorithmique n'est pas triviale pour un niveau de licence.

4.5. Approche utilisée

Algorithme Negamax avec élagages Afin d'obtenir un maximum de résultat dans la phase de l' analyse, nous devons avoir un algorithme rapide. L'algorithme Monte-Carlo pourrait être une solution sur la rapidité, avec l'expérience du groupe, nous avons choisi l'algorithme paranoïde sous forme négamax qui semble être le plus facile à réaliser dans le temps de ce projet. Pour optimiser le temps de calcul, nous avons rajouté quelques élagages pour l'accélérer.

Élagage Alpha-Bêta L'élagage alpha bêta est une technique permettant de couper les branches de l'arbre qui représente des sous-arbres possédant des valeurs qui ne contribuent pas au calcul minmax de notre algorithme.

4.6. Remarques sur les résultats obtenus

Aucun déplacement trouvé Lorsque l'algorithme paranoïde perçoit la fin de son équipe dans toutes ces branches, vu que l'heuristique est une évaluation par rapport au terrain (en effet vu que l'équipe est morte l'heuristique renverra 0). Il sera incapable de faire la différence entre ces états menant vers sa fin, l'algorithme renverra la première action, donné dans la liste d'actions possible. Pour éviter ce genre d'événement, nous avons ajouté à l'algorithme une détection d'action permettant de déterminer si l'algorithme n'a pas trouvé de solutions. L'algorithme choisira un coup aléatoire parmi les actions possibles, qui ne tuera pas le joueur.

Phénomène d'horizon Nous avons observé un phénomène lors de nos tests de l'algorithme paranoïde. Le joueur seul pense que la partie est finie et passe dans le mode survie de l'algorithme expliqué ci-dessus*, alors qu'il pouvait s'en sortir.

En effet nous avons vu que l'une des faiblesses de cet algorithme est que nous ne sommes pas garantis de la fiabilité du coup choisie au tour suivant. Vu que le joueur seul

a une vision plus profonde de la partie que ses adversaires, il perçoit l'opportunité de ses adversaires de le tuer. Mais si cette opportunité est vue dans une profondeur plus grande que celle de l'adversaire alors elle n'est pas garantie d'arrivée, car l'adversaire jouera le meilleur coup à hauteur de sa profondeur. Ce phénomène fournit une preuve de la limite de puissance de l'algorithme paranoïde.

Nous avons vu l'impact le plus gênant de ce phénomène, mais nous pouvons aussi remettre en cause avec le même principe, tout les calculs pris en compte lorsque la profondeur d'un joueur dépasse celle de son adversaire. En effet lorsque les calculs sont faits dans des hauteurs de profondeur accessible par tous les joueurs, les joueurs percevront les états futurs. En revanche dès que la profondeur dépasse la profondeur de l'adversaire, le joueur obtient une prévision des meilleurs états futurs de son adversaire. Il jouera le meilleur coup possible en fonction de ces états, mais ce coup n'étant pas calculé dans les autres états futurs possibles, ce coup ne sera plus le meilleur coup possible (remarque : dans la majorité des cas il reste un coup assez viable).

4.7. Pistes d'améliorations

Concernant l'optimisation de temps d'éxecution Lorsque l'algorithme détecte à hauteur de sa profondeur que la partie est finis, il arrête de jouer le meilleur coup possible et joue des coups aléatoires. Après chaque coup l'algorithme est rappelé, ce qui ralentit grandement l'analyse de résultat. Nous pourrions retirer le mode survie de l'algorithme, et déclarer un vainqueur par abandon. Mais nous avons vu que l'algorithme peut se tromper lorsque sa profondeur est plus grande que l'adversaire.

Nous avons observé plusieurs fois des situations où l'algorithme prévoit qu'il va perdre. Ces prévisions ont 75% d'être vrai.

Conclusion, si nous utilisons cette amélioration nous perdrons en précision de résultat, mais nous gagnerons plus en temps d'exécution de l'analyse.

Concernant le phénomène d'horizon Il existe l'algorithme Expectimax qui va jouer des coups aléatoires et faire une moyenne. Nous pourrions garder notre algorithme paranoïde, puis lorsque la profondeur dépasse celle de l'adversaire alors les coups des adversaires seront joués aléatoirement et une moyenne sera faite à partir de ce niveau.

5. Intelligence Artificielle - Heuristique

5.1. Éléments techniques

Heuristique (4.1.3.) : Une heuristique permet de faire une évaluation d'un état, afin de différencier un état d'un autre. Ces différences permettent aux algorithmes de faire un choix entre plusieurs actions.

5.2. Nécessités

Évaluation du plateau Lors d'une partie de jeu Tron, le joueur qui possède la plus grosse zone de contrôle sur le plateau gagnera. En partant sur cette stratégie, l'heuristique devra calculer la zone de contrôle des joueurs.

5.3. Problème

Comment pourrions-nous faire une heuristique rapide, mais aussi donnant des résultats fiables ? L'heuristique devra évaluer la valeur d'une situation. Devant parcourir tout le plateau, pour évaluer un état, il faudra beaucoup de calculs pour obtenir une valeur précise. Il nous faudra être suffisamment rapides sur cette partie afin de générer assez de tests pour l'analyse des configurations de la partie.

5.4. Approche possible

Parcours de recherche BFS pour un joueur Ce parcours va calculer la distance de chaque case depuis la position du joueur, cette opération est répétée pour tous les joueurs. L'attribution de la case est donnée à l'équipe possédant la plus petite distance sur cette case. La complexité de l'heuristique est de n * m * c.

Parcours de recherche BFS joueur par joueur Dans cette approche, le parcours BFS va s'étendre à partir de tous les joueurs. La complexité de l'heuristique est de n*m.

Évaluation de la zone de contrôle en nombre de cases possédé lorsque nous évaluons un plateau en nombre de cases, minimiser signifie réduire la taille de la zone du terrain contrôlé par le joueur. Sauf que réduire la zone adversaire ne signifie pas forcément augmenter la sienne. L'algorithme paranoïde minimise la zone du joueur lors de la simulation du coup de son adversaire, alors que l'adversaire à son tour essayera de maximiser sa zone. Nous nous retrouvons dans un état qui n'a pas été prévu par l'algorithme et donc fausse toute la perception des états futurs calculée par l'algorithme.

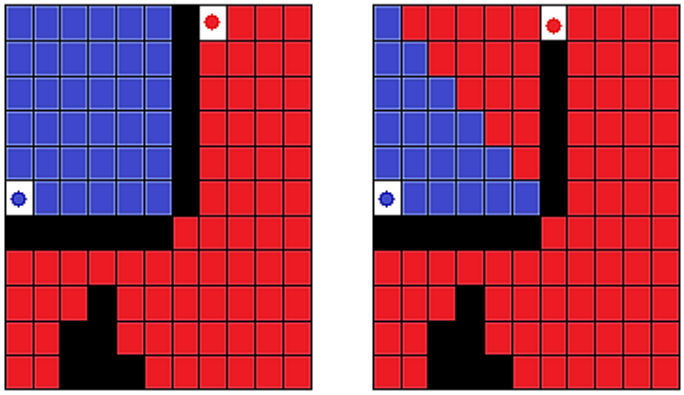


FIGURE 5.1. – bleu = 20 cases/ 20% ; rouge = 82 cases/ 80%

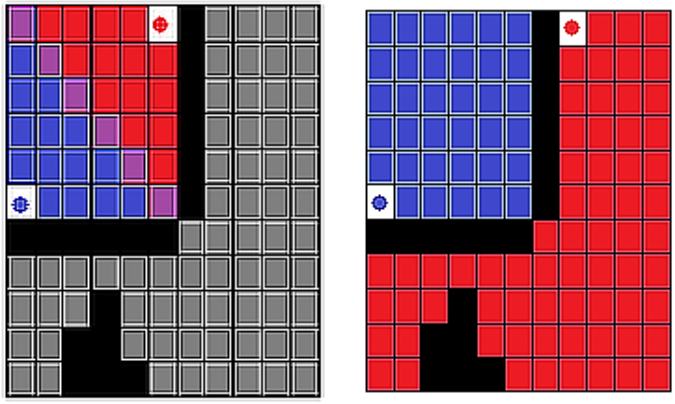


FIGURE 5.2. – bleu = 14 cases/ 50% ; rouge = 14 cases/ 50%

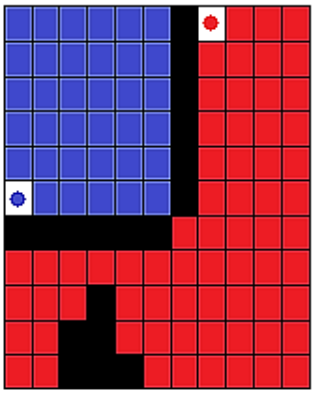


FIGURE 5.3. – bleu = 35 cases/ 35% ; rouge = 66 cases/ 65%

Évaluation de la zone de contrôle en pourcentage du terrain jouable possédé L'évaluation d'une zone contrôle convertit en pourcentage de contrôle du terrain jouable, permet d'obtenir une heuristique à sommes nulle . En effet lorsqu'un joueur maximise son pourcentage de contrôle du terrain, il diminuera celui de son adversaire et vice-versa.

5.5. Approche utilisée

Parcours de recherche et Optimisation Il est évident que nous allons utiliser l'approche avec la meilleure complexité. Cette méthode est rapide pour sa fiabilité, mais lors de l'exécution de l'analyse de donnée, cette partie est la plus coûteuse de notre simulation.

Notre principal objectif est d'obtenir suffisamment de résultats, afin de déduire qu'elles seront les paramètres idéaux pour résoudre la problématique. Avec notre calcul de la zone de contrôle, les estimations de calcul s'élèvent à plus de 48H.

Pour améliorer le temps de calcul de la zone de contrôle, on suppose que le joueur seul possède moins de terrain que la coalition et qu'une zone qu'il ne contrôle pas est une zone appartenant à la coalition.

Avec ces suppositions on remarque que les cases isolées (voir case grise de l'image tag 2) n'appartiennent à aucune équipe. Avec cette supposition on constate qu'une case que ne possède pas le joueur seul n'est pas forcément une case appartenant à son adversaire. Utiliser ce principe nous ferait gagner en temps de calcul, mais nous perdrons en fiabilité.

Pour générer cette amélioration, l'heuristique calculera la zone de contrôle de chaque équipe, le parcours BFS sera interrompu lorsque le joueur seul ne peut plus obtenir de case pour sa zone de contrôle. En effet avec cette solution la zone de contrôle des joueurs de la coalition risque de ne plus être calculée en entier. Pour obtenir la zone de contrôle de la coalition, une approximation de celle-ci va s'ajouter aux calculs de l'heuristique.

À partir de la zone de contrôle du joueur seul et la taille du terrain jouable (le nombre de cases libre), nous pouvons en déduire une approximation de la zone contrôlée par la coalition. (ex. : zone contrôle seule = (cases possédées / terrain jouables) * 100 zones contrôle coalition = 100 - zone contrôle seule)

> Dans cette approche, nous ne sommes plus obligés de calculer toutes les cases du terrain jouable. En effet toutes les cases d'une zone ne sont plus calculées lorsqu'une zone est plus grande que la zone du joueur seul.

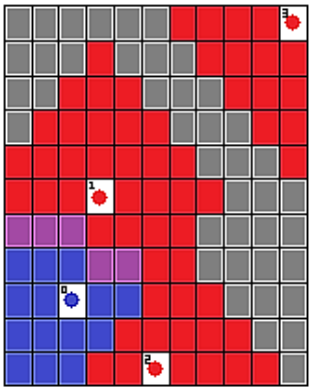


FIGURE 5.4. – Calcul de la zone de contrôle du joueur 3

Avec cette méthode, nous gagnons en temps d'exécution et nous perdons en fiabilité (à cause des cases isolée et constatée). Le moyen d'augmenter la fiabilité serait d'améliorer l'approximation, mais sans calculer la zone de contrôle de la coalition. L'estimation de calcul s'élève désormais à 24H, ce qui est toujours élevé.

On remarque qu'un joueur trop éloigné n'aura pas d'impact sur le calcul de la zone de contrôle du joueur seul. On considéra un joueur trop éloigné lorsqu'il est éloigné sur

une distance de deux fois supérieure à celle de la profondeur du joueur seul. Avec une distance deux fois supérieure à celle de la profondeur du joueur seul, même si les deux joueurs se rapprochent tout le long des tours, la distance est telle, qu'à la fin de la recherche ils ne pourraient toujours pas interagir l'un avec l'autre.

Vu que l'heuristique va calculer une zone qui n'a pas d'impact sur la zone de contrôle du joueur seul, nous pouvons supprimer tous les joueurs qui sont trop éloignés du joueur seul. En effet au début du calcul de l'heuristique, nous initialisons la position de départ de chaque joueur. Grâce à la distance de Manhattan entre les joueurs, nous pouvons déterminer les joueurs assez proches et les prendre en considération lors du calcul des zones.

> Perde en fiabilité n'est pas forcement un point négatif lors de la phase d'analyse. En effet grâce à l'optimisation des temps de calcul nous pouvons générer beaucoup plus de résultats. Parmi ses résultats les petites erreurs de fiabilité seront noyées dans la masse d'information, car nos erreurs de fiabilité ne sont pas une généralité, mais des cas très particuliers de partie.

5.6. Remarques sur les résultats obtenus

Maximiser, minimiser un nombre de cases Avec une heuristique basée sur les nombres de cases contrôlées par une équipe, nous avons constaté que le joueur seul essayait d'augmenter sa zone alors que les joueurs de la coalition essayaient de réduire la taille du joueur seul, dans l'algorithme paranoïde la stratégie des dans camps doit être identique, ce qui n'est pas le cas avec une telle heuristique.

Pistes d'amélioration Une autre solution serait de modifier l'heuristique en ajoutant à son évaluation d'un plateau, le nombre de joueurs présents sur le plateau, le nombre de tour, la taille de la zone de contrôle du terrain adverse. Ce qui permettrait de rajouter plus d'évaluation sur un état lorsqu'un joueur possède une zone de contrôle faible.

6. Analyse - Exploration

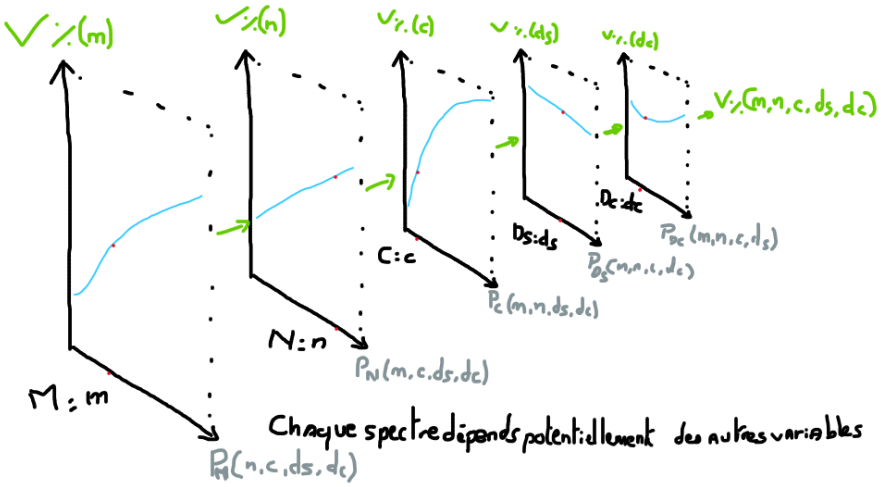
6.1. Nécessités

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 ?

6.2. 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 ?



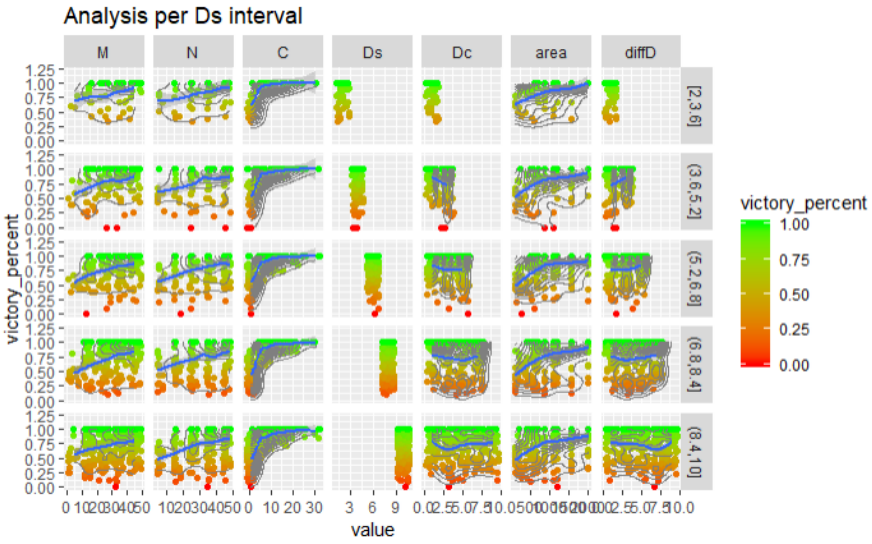
> Comment pourrions-nous analyser nos données pour pouvoir y déceler des informations de la façon la plus efficace et complète possible sur autant d'axes ?

6.3. 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.



6.4. 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.

6.5. 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 paraitraît 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

7. Analyse des données de l'IA

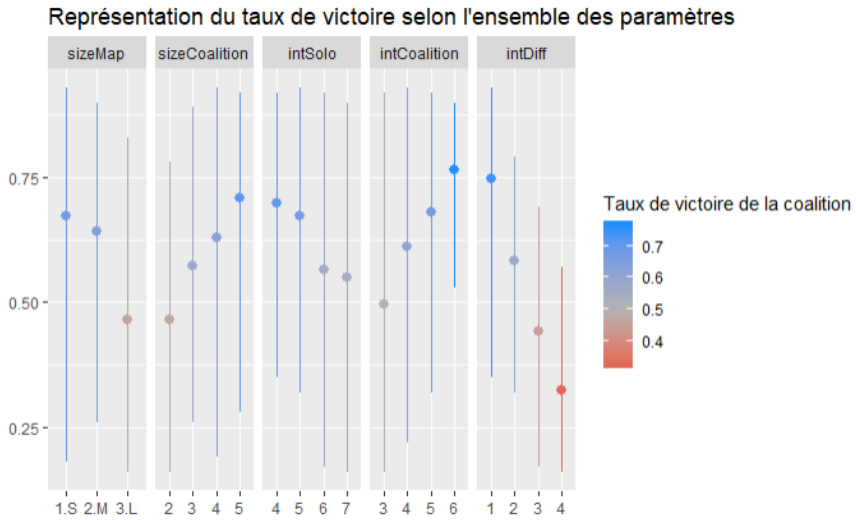
Paramètres initiaux Cette simulation as été réalisée avec 100 réitérations pour chaque combinaison possible avec les paramètres suivants.

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


Time estimated 0d 3h 33m 41s |[##########]----------------------------------| 25.8%     @ 1.44s/game    curr (1.S,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.

7.1. Analyse d'ensemble

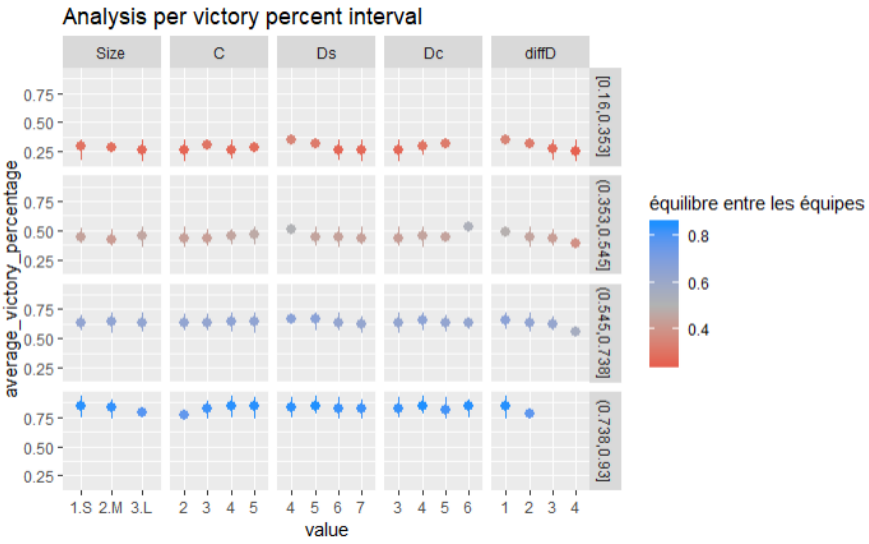


Des tendances variées Nous pouvons d'abord remarquer la présence de cinq tendances générales de données :

- 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.

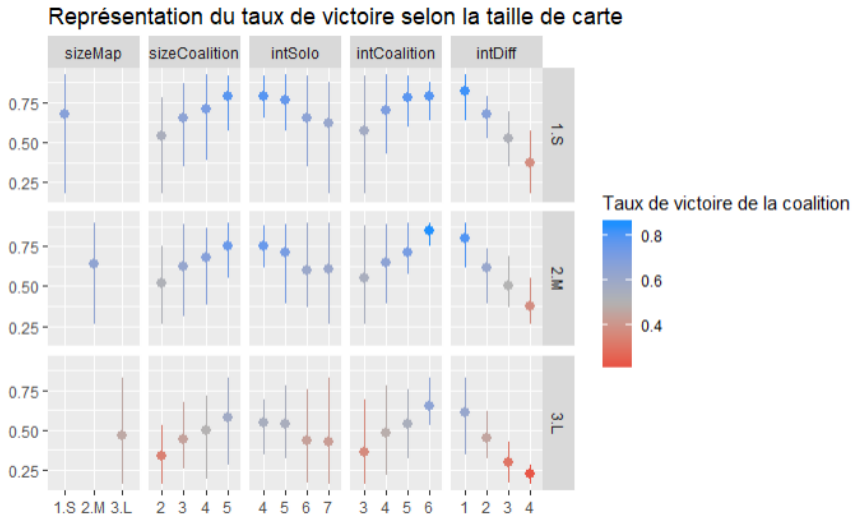
7.2. Analyses détaillées

7.2.1. 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. Les cartes plus petites semblent aussi être un avantage pour la coalition.

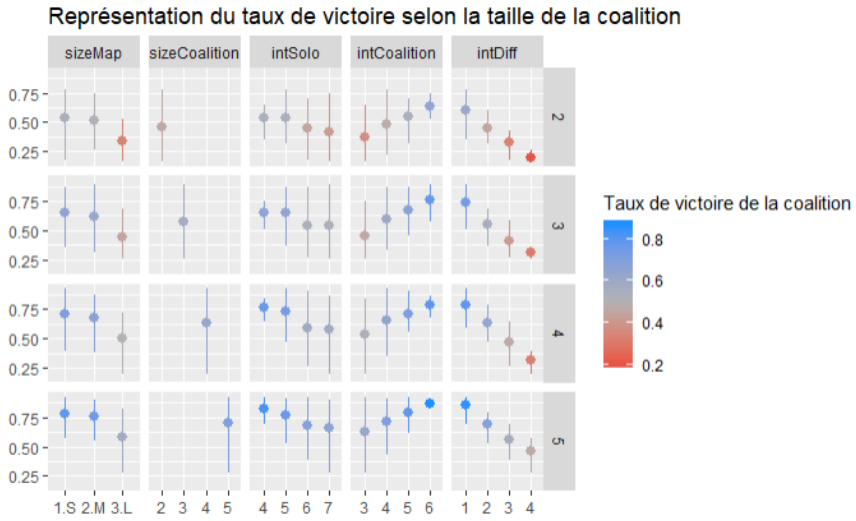
7.2.2. Découpage du spectre selon des intervalles de la taille de la carte



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.

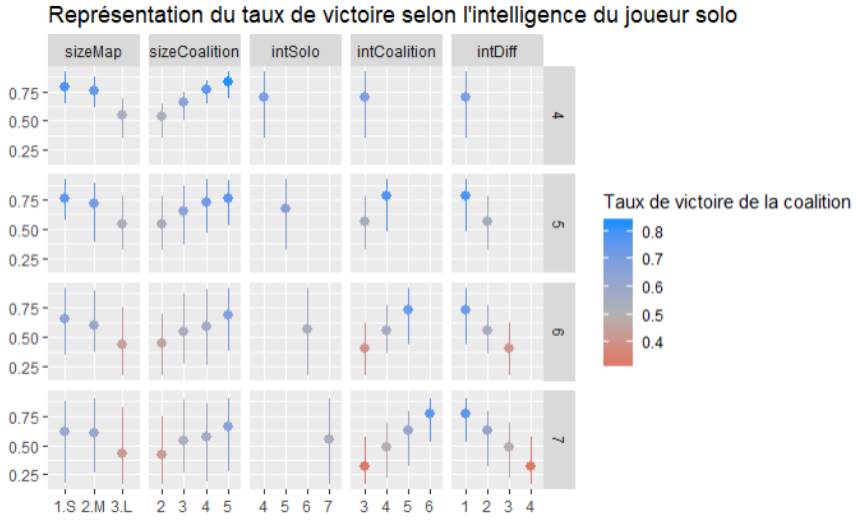
En revanche, les grandes cartes semblent avoir un effet négatif sur les chances minimales de victoire sur tous les facteurs.

7.2.3. Découpage du spectre selon des intervalles de C



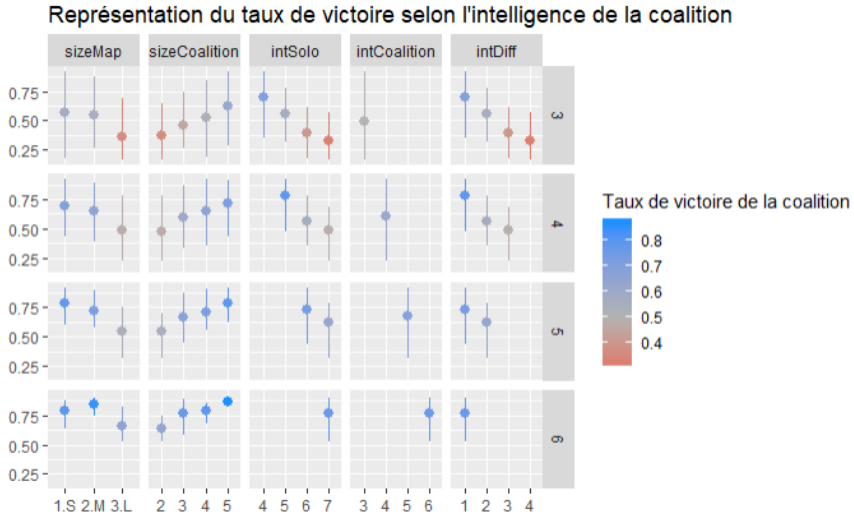
Le surnombre semble avantageux Les chances de victoire générales semblent augmenter avec de plus grandes valeurs de C de façon homogène.

7.2.4. Découpage du spectre selon des intervalles de Ds



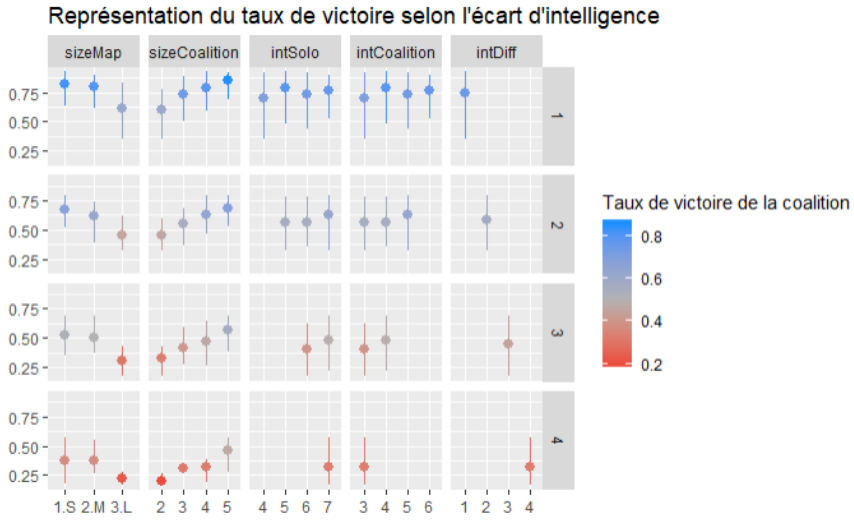
Un joueur solo intelligent est plus difficile à battre Nous pouvons remarquer que de plus grandes valeurs de Ds semblent diminuer les chances générales de victoires.

7.2.5. Découpage du spectre selon des intervalles de Dc



Une coalition intelligente bats le joueur plus facilement Inversement, les plus grandes valeurs de Dc semblent maximiser les chances générales de victoire.

7.2.6. 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 ($diffD = Ds - Dc$).

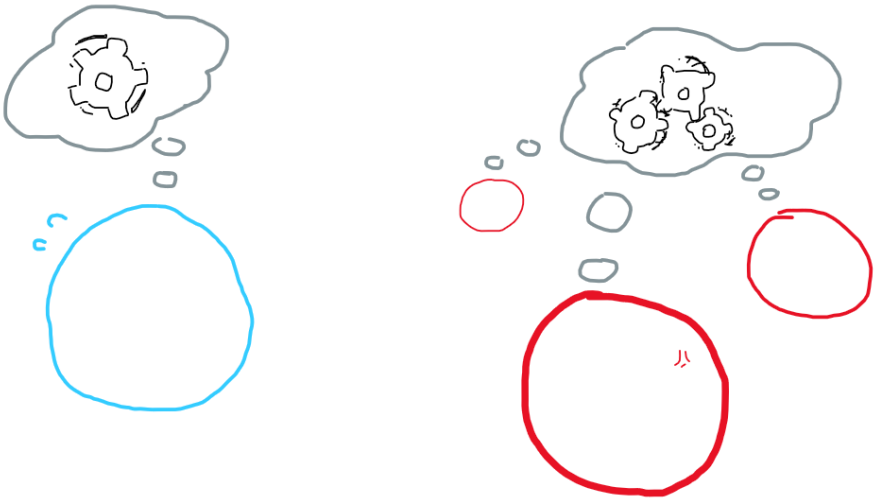
7.2.7. Conclusions d'analyse

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 as 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 plaffonner plus vite si l ecart 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.



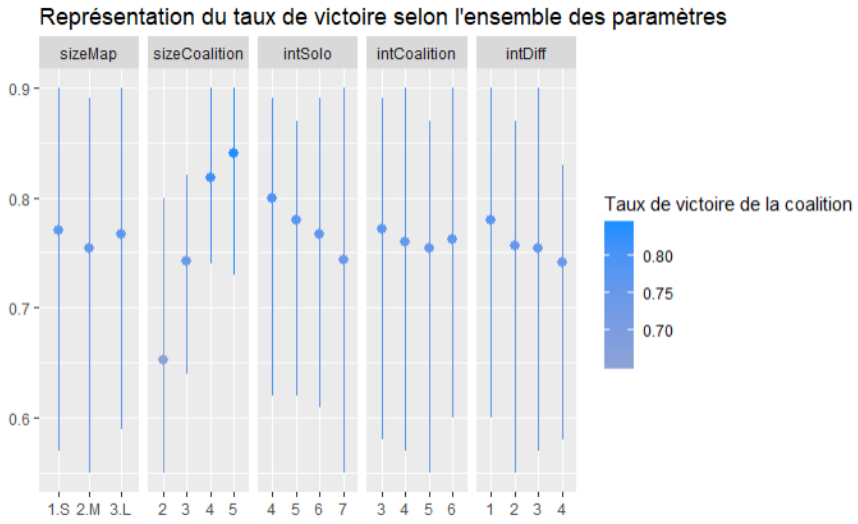
8. Analyse des données du modèle

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: 8×8
Largest map dimensions: 22×22

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

8.1. Analyse d'ensemble

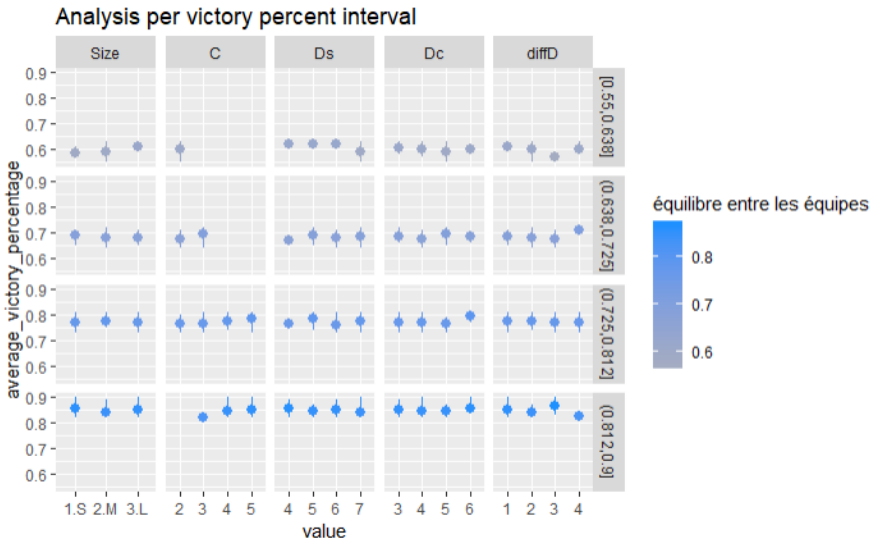


Un seul facteur sort du lot À commencer par le facteur C qui est le seul avec semble vraiment avoir de nette relation avec le pourcentage de victoires.

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

8.2. Analyses détaillées

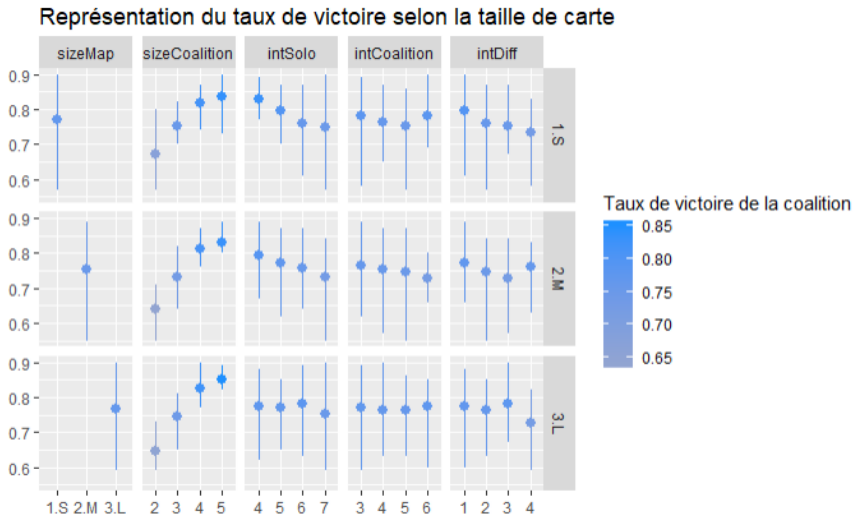
8.2.1. Répartitions des pourcentages de victoires



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

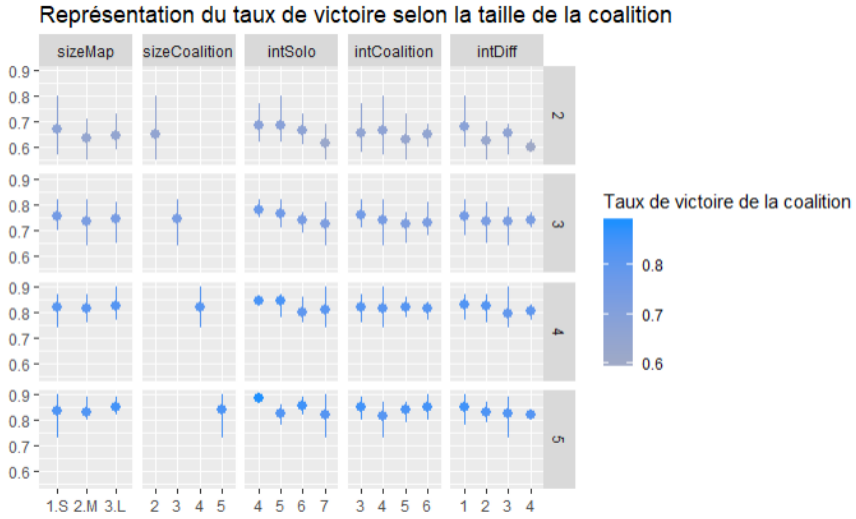
Les autres facteurs restent relativement homogènes, ceux ci ne semblent donc pas véritablement influer la fin des parties.

8.2.2. Découpage du spectre selon la taille de la carte



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

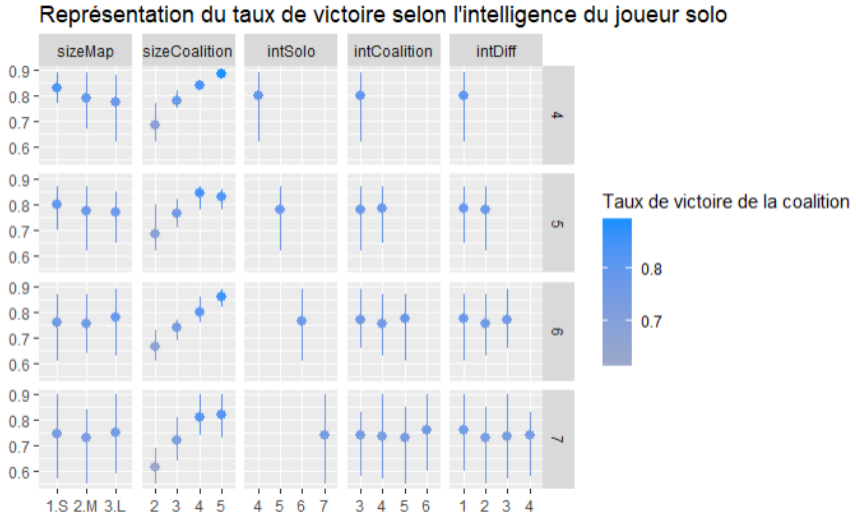
8.2.3. Découpage du spectre selon des intervalles de C



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.

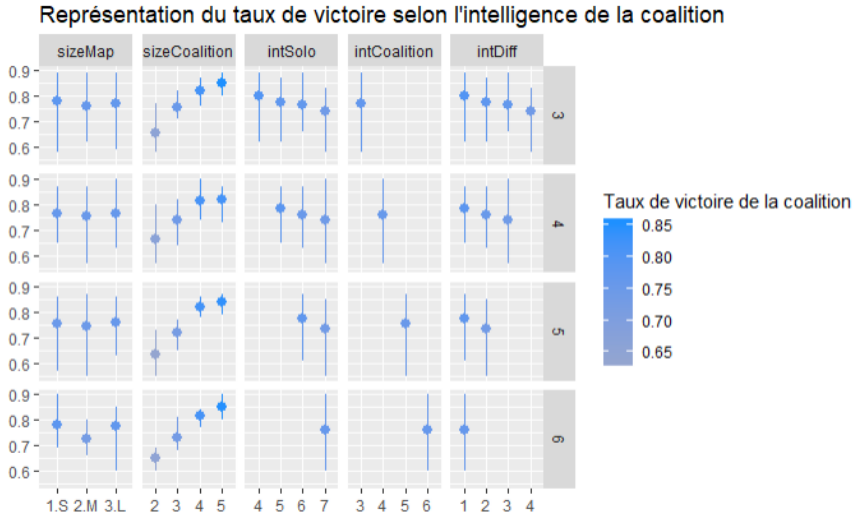
8.2.4. Découpage du spectre selon des intervalles de Ds



Une légère diminution des chances de victoires Ds ne semble vraiment pas influer 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.

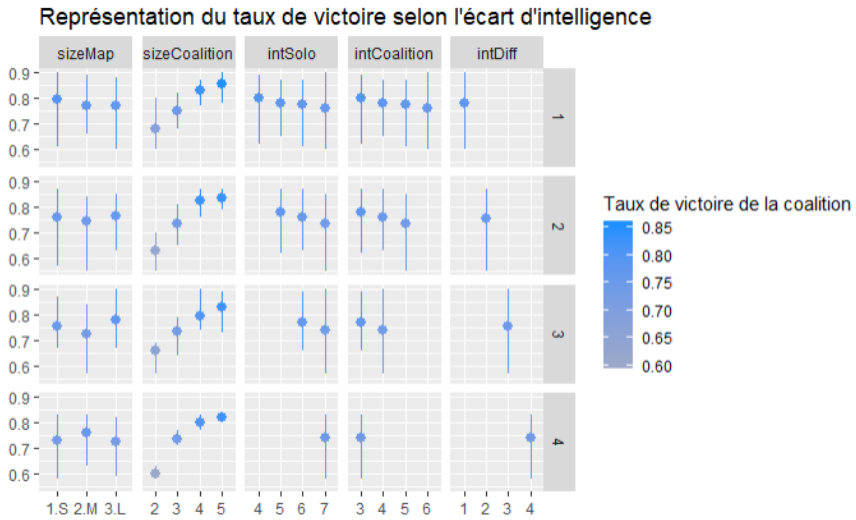
À plus grandes valeurs de Ds, les probabilités de vitoires semblent plus nuancées et défavoriser la coalition.

8.2.5. Découpage du spectre selon des intervalles de Dc



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

8.2.6. 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 influer particulièrement sur les chances de victoires. Il ne semble donc pas y avoir de véritable interaction entre les équipes.

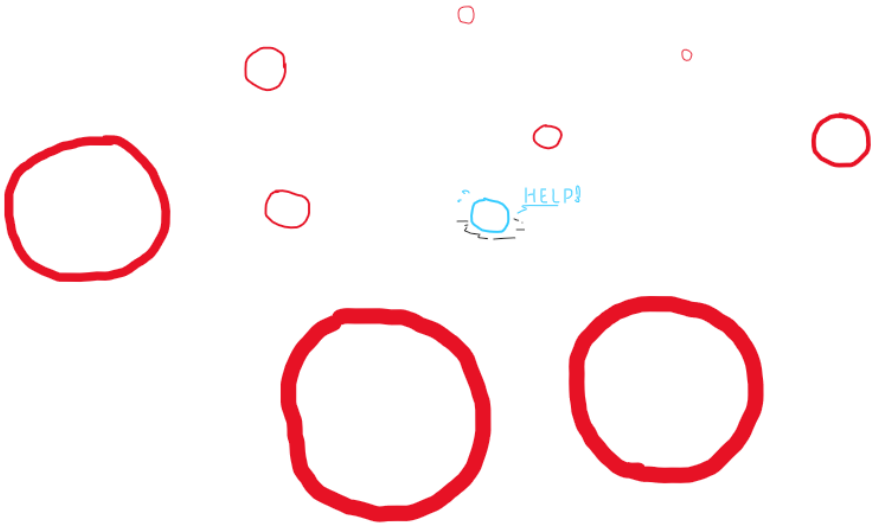
> Ce modèle ne colle malheureusement pas au comportement complet de notre intelligence artificielle, mais permet d'approximer à moindres coûts une stratégie de survie pure, comme lorsqu'un joueur est bloqué dans une zone sans adversaires.

8.2.7. 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 as 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.



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

Sixième partie

Problèmes, tests et expérimentations

9. Conclusion

Ce chapitre est actuellement en cours d'écriture.

9.1. Résumé des objectifs au résultat final

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.

Citons les résultats finaux de nos analyses : D'après nos analyses finales sur un échantillon de 12000 simulations, nous avons pu déterminer que :

- La pire coalition possible serait une coalition qui joue quasiment au hasard, avec peu d'équipiers et sur une grande carte.
- La coalition idéale semble être une coalition en surnombre, d'intelligence égale au joueur solo, et sur une petite carte.

De plus, sur toute carte moyenne ou petite, la coalition est statistiquement meilleure.

9.2. Enrichissement personnel

9.2.1. Vincent :

Ce projet m'as permis de découvrir plusieurs phénomène sur l'algorithme MinMax. J'ai beaucoup appris sur l'impacte que peut avoir une heuristique sur un algorithme. J'ai pu me documenter sur plusieurs autres algorithmes. Réaliser tout ces optimisations pour le modules d'analyse, m'a fait comprendre que l'on peut passer par d'autre chemin pour un résultat identique mais plus rapide. Travailler avec ce groupe m'as permis d'apprendre de nouvelle méthode de travail, et aussi des nouveaux outils en informatique.

9.2.2. Alexis :

À remplir

9.2.3. Christopher :

Ce projet m'as beaucoup appris, en très peu de temps j'ai pu me familiariser avec les particularités de cython, et expérimenter avec. Les nombreuses fouilles à l'optimisation ont été un bonheur de problèmes à creuser et les accélérations progressives encourageaient à toujours creuser plus loin.

De même, m'étant familiarisé avec R aux alentours de Décembre 2018 pour des projets extra-scolaires, le travail sur la représentation finale des données fut long et riche en apprentissages. De nombreux volontaires ont participé aux critiques sur les différentes itérations de celles ci pour améliorer la compréhensibilité ainsi que la clarté en recherchant jusqu'aux couleurs visibles pour des daltoniens, à la recherche des données les plus pertinentes et intuitives sur l'intégralité de nos données, et un travail de représentation de celles ci pour en rendre la lecture la plus naturelle possible.

Nous espérons que la représentation finale y fasse honneur.

9.2.4. Walid :

À remplir

9.3. Perspectives envisagées, appréciation perso, poursuite..

9.3.1. Vincent :

Nous pourrions utiliser l'algorithme Monte-Carlo pour gagner du temps de calcul. Ce qui permettrait de supprimer l'heuristique, qui possède une très grande importance dans l'algorithme. Imaginer une solution sans heuristique permettrait d'enlever plusieurs problème.

Mener une étude sur les positions idéal que pourrait avoir les joueurs sur le plateau serait très enrichissant.

9.3.2. Alexis :

À remplir

9.3.3. Christopher :

En travaillant sur ce projet, notre groupe s'est souvent repenché sur le sujet de l'approche de Monte Carlo. La voyant repasser depuis quelques années déjà au cours des recherches sur divers projets, et ayant étudié les explications sur le concept via sa page wikipédia anglophone, j'aimerais beaucoup tenter de l'appliquer lors d'un projet futur pour expérimenter avec et vraiment être certain de saisir le concept.

Les optimisations semblent aussi être un domaine fascinant, et je creuserais très certainement le concept dans le futur.

Sur un autre sujet, cette équipe est probablement la meilleure équipe avec laquelle j'aie pu travailler : motivée et intéressée. J'espère sincèrement pouvoir retrouver tout le monde en master l'année prochaine et avoir l'occasion de refaire des projets de ce style plus souvent !

9.3.4. Walid :

À remplir

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 detection d'anomalies de tenter de trouver un modèle mathématique qui décris bien l'évolution de notre système.

Mais si les calculs d'heuristiques 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 reccurentes.

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 pouvont 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 corresponds 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 meme genre de résultats que le calcul complet de notre phénomène de départ.

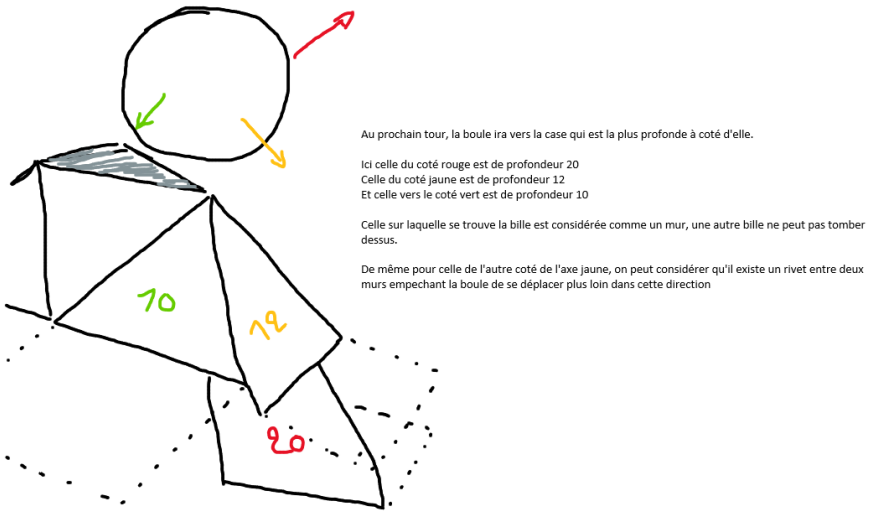
De plus, cela permettrais aussi potentiellement de mettre en équation le comportement des acteurs de notre phénomène, ici les IA, et de potentiellement trouver un modèle simplifié et fonctionnel pour leur heuristique.

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.

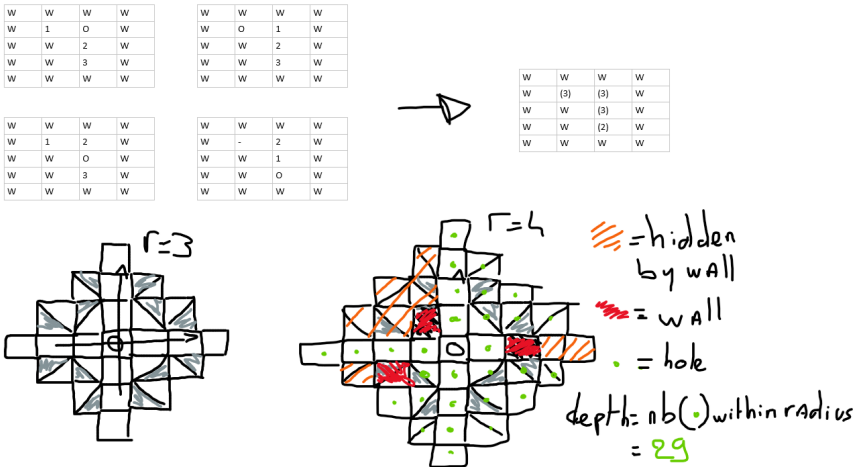
10.4. Approche utilisée

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.



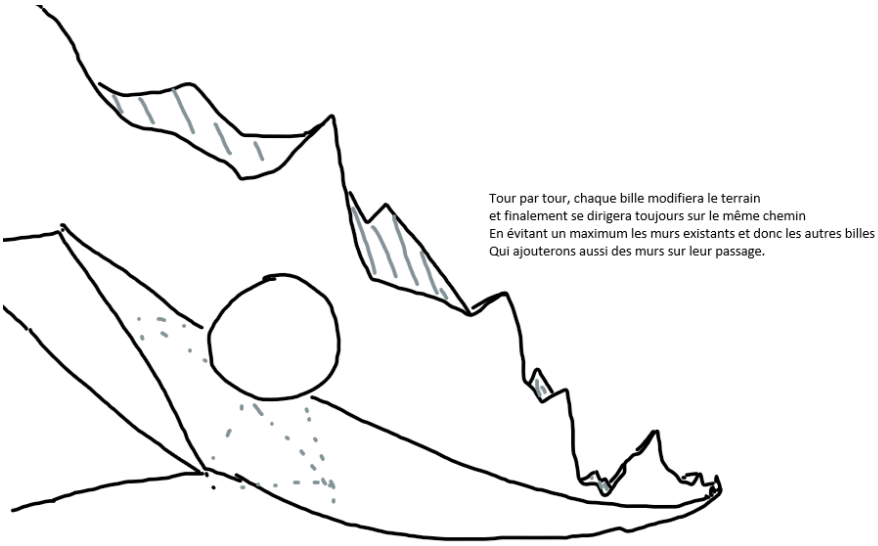
Nous avons donc pris les joueurs pour des billes (littéralement), et les considérons désormais en roulis perpétuel vers la pente la plus accentuée qui s'offre à eux.

À la suite de leur roulis (changement de case), un mur se crée à leur position actuelle, les forcant 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'as plus de pente sur laquelle rouler. Elle est alors retirée du jeu et considérée hors jeu.



Ce système devrais favoriser le contrôle de la carte de façon naturelle, car les billes seront donc naturellement inclines 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 as 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 un différence plus porté sur l'écart d'intelligence que sur le reste.

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

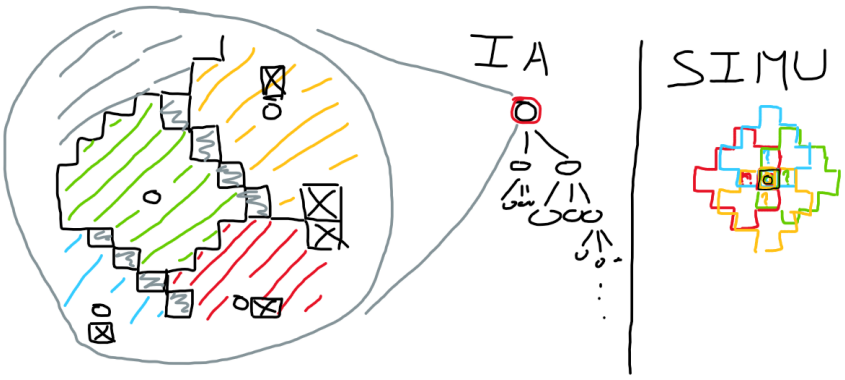
> Il serait peut être possible cependant d'utiliser notre modèle en optimisation pour les IA en totale autarcie et économiser en temps de calcul quand aucune stratégie offensive n'est désormais utile.

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 $3^C * Ds$ fois dans le pire des cas (tous les joueurs peuvent aller dans 3 directions à chaque tour, sur Ds 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.

$$D = \frac{C * Dc + 1 * Ds}{C + 1} \tag{10.1}$$

$$\approx 9 \tag{10.2}$$

$$t_{max} = \frac{M * N}{C + 1} \tag{10.3}$$

$$\approx 68 \tag{10.4}$$

10.6.1. Modèle IA

$$nbCases(t, M, N, C) = M * N - (C + 1) * t \tag{10.5}$$
$$nbCases(t) = 2500 - 37 * t \tag{10.6}$$
$$nbCases/tour/joueur(t, D) = \int_t^{t+D} nbCases(x)dx \tag{10.7}$$
$$= [2500 * t - 37 * \frac{t^2}{2}]_t^{t+D} \tag{10.8}$$
$$= 2500(t + D - t) - 37 * \frac{(t + D - t)^2}{2} \tag{10.9}$$
$$nbCases/tour/joueur(D) = 2500 * D - \frac{37}{2} * D^2 \tag{10.10}$$
$$totalCases(t_{max}, C, D) = (C + 1) * t_{max} * nbCases/tour/joueur(D) \tag{10.11}$$
$$= (C + 1) * t_{max} * (2500 * D - \frac{37}{2} * D^2) \tag{10.12}$$
$$= 37 * 68 * (2500 * 9 - \frac{37}{2} * 9^2) \tag{10.13}$$
$$= 2516 * (22500 - 18.5 * 9^2) \tag{10.14}$$
$$= 2516 * (22500 - 1498.5) \tag{10.15}$$
$$= 2516 * 21001.5 \tag{10.16}$$
$$totalCases(t_{max}, C, D) \approx 52839774 \tag{10.17}$$

10.6.2. Modèle simulé

$$nbCasesInRadius(D) = \int_0^D 4 * x dx \tag{10.18}$$

$$= [4\frac{x^2}{2}]_0^D \tag{10.19}$$

$$nbCasesInRadius(D) = 2D^2 \tag{10.20}$$

$$totalNbMurs(t, C) = t * (C + 1) \tag{10.21}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$= 17982 * (68 - 34) \tag{10.39}$$

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

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.

$$\frac{totalCases_{IA}}{totalCases_{simu}} = \frac{52839774}{611388} \tag{10.41}$$

$$\frac{totalCases_{IA}}{totalCases_{simu}} \approx 86 \tag{10.42}$$

> 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.