UNIVERSITÉ DE LORRAINE | IUT nancy Charlemagne

Présentation du sujet de projet tutoré:

Simulation de poursuite-évasion collaborative et compétitive entre agents informatiques intelligents

Tuteur: Guénaël Cabanes

Matias Amaglio
Maëlle Bitsindou
Luc Dechezlepretre
Célie Ponroy
Année universitaire 2024- 2025

Table des matières

I. Description détaillée du sujet

A. Contexte

- Langage utilisé:

D'après nos recherches les langages les plus utilisés dans le domaine des jeux vidéos sont le C++ et le C#. Et Python est le langage le plus utilisé dans le développement en Intelligence Artificielle.

Malgré cela nous allons utiliser **Java** lors de notre projet tutoré. Java est le langage que nous connaissons le mieux, ce qui simplifiera les choses.

- But de la simulation:

Pour un meilleur déroulement de la simulation, nous avons fixé quelques règles précises. Il faut savoir que dans cette simulation, il y aura 2 ou plusieurs agents: des gardiens et des prisonniers.

La simulation s'arrête lorsque chaque prisonnier de la partie a un statut:

- soit il est attrapé
- soit il a réussi à s'échapper

Le nombre d'agents pourrait être amené à évoluer pour complexifier l'expérience.

- Détails des comportements de chaque agent:

Chaque agent (prisonnier et gardien) a un champ de vision circulaire autour de lui en prenant en compte les différents obstacles (murs…). Chaque agent aura également connaissance de toute la carte et de ses sorties. Ce dernier point pourra évoluer au cours de l'avancement du projet.

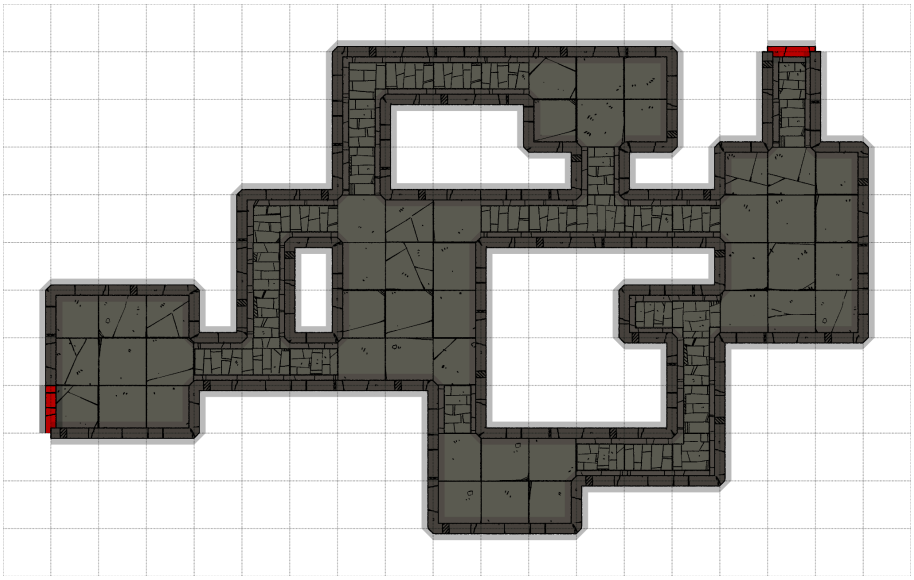
Chaque agent pourra également effectuer un déplacement par unité de temps, ce qui rendra le jeu plus fluide.

- Détails de la carte:

La carte sera constituée d'un système de grille, ce qui permettra de développer plus simplement la vision et les déplacements des différents agents.

Dans un premier temps, la carte sera fixe, elle pourra varier une fois l'apprentissage terminé. Nous pourrons ensuite analyser les résultats dans différents environnements.

Une carte est constituée de salles, couloirs et de sorties. Il y a une sortie de plus que de gardien. Les différents éléments composant une carte pourront être un facteur d'évolution au cours du projet. (obstacles, cachettes, etc…)



Maquette représentant une carte quadrillée

- Moteur de jeu:

Avant de faire un choix sur le moteur de jeu utilisé, nous avons commencé par faire des recherches préalables.

Il y a plusieurs solutions telles que **Unity et Godot** qui ne sont pas adaptées car ce sont des applications et elles utilisent le "no code". Nous nous sommes ensuite intéressés aux librairies, principalement en java comme **LibGDX, Pygame4J, Cocos2d,** etc. Ces librairies sont intéressantes mais nous avons considéré ces ajouts comme non nécessaires. Nous voulons nous concentrer sur la fonctionnalité de notre application notamment la partie algorithmique et nous voulons pouvoir avoir un contrôle total sur notre affichage . Nous avons finalement décidé de nous baser sur les travaux de Monsieur Vincent Thomas (voir annexe sources). Le moteur de jeu sera alors fait en **Java et JavaFx**.

B. Missions attendues

Ce projet peut être décomposé en plusieurs tâches principales, notamment :

- **Développer un environnement visuel** en 2D représentant la prison avec divers éléments tels que des obstacles, des murs et des sorties, ainsi qu'une interface avec un ou plusieurs utilisateurs pouvant prendre le contrôle de certains agents.
- **Créer et implémenter les comportements** automatiques des deux types d'agents avec des objectifs et des stratégies distinctes. Nous explorerons par exemple des mécanismes d'inférence bayésienne pour créer pour chaque agent une "carte mentale" probabiliste de la position présumée des autres agents.. Cette carte sera couplée, dans un premier temps, avec des arbres de décisions comportementaux qui dicteront le comportement des agents en fonction de leur "carte mentale".
- **Concevoir des outils d'analyse** de la performance des agents, permettant de faciliter les comparaison entre différentes stratégies implémentées.
- **Mettre en place un système d'apprentissage automatique** pour optimiser les prises de décision des agents :

    - Intégrer des mécanismes d'apprentissage par renforcement pour permettre aux agents d'apprendre et d'optimiser leurs stratégies respectives en fonction de récompenses et des pénalités reçues en fonction de leurs performances.
    - Tester différents algorithmes pour d'apprentissage à la place ou en complément de l'apprentissage par renforcement, et comparer les différents comportements résultant.
    - Faire varier l'environnement d'apprentissage en ajoutant différentes règles et obstacles

II. Recherches des différents types d'algorithmes

A. Inférence bayésienne

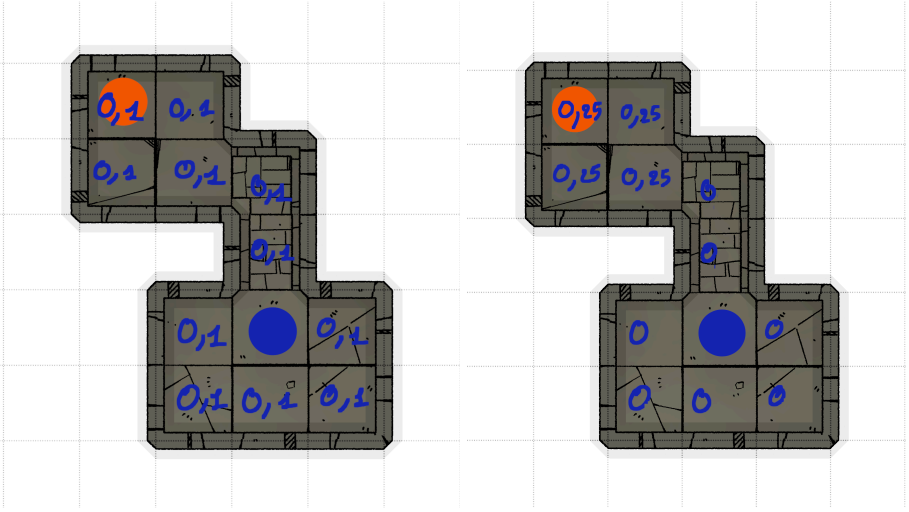
1. Étude technique

Avant d'aborder les méthodes d'apprentissage, nous allons développer un algorithme d'inférence bayésienne qui constitue la fondation des différents algorithmes d'apprentissage. L'inférence bayésienne est une méthode d'inférence statistique visant à calculer le degré de confiance à accorder à une hypothèse en fonction des données disponibles. Elle repose sur le théorème de Bayes, qui établit les principes de calcul d'une probabilité conditionnelle.

Trois principes fondamentaux caractérisent la méthode d'inférence bayésienne :

1. La connaissance antérieure ou a priori est intégrée dans le calcul de la probabilité.
2. Chaque nouvelle donnée permet de réévaluer l'hypothèse en fonction des connaissances antérieures.
3. Le calcul est itératif, c'est-à-dire que, à chaque nouvelle information, le calcul est révisé et les résultats sont réévalués en fonction de ce niveau de connaissance antérieure.

Appliquée à notre projet, l'inférence bayésienne permettra aux différents agents de déduire la position des autres agents. Et ainsi définir leurs stratégies en fonction de leur comportement.

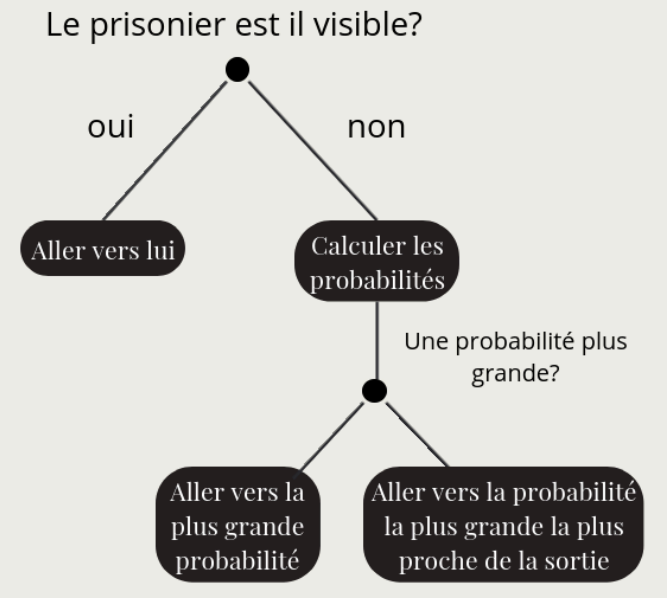


Maquettes de cartes avec inférence gardien visible avant et après avoir reçu une nouvelle donnée

B. Arbre de décision

1. Étude technique

L'arbre de décisions est une manière simple de prendre des décisions. Il est représenté sous la forme d'une structure arborescente où chaque nœud interne représente un test sur un attribut, chaque branche correspond à un résultat de ce test et chaque feuille de l'arbre représente une action.



Exemple d'arbre de décision sur notre projet

L'arbre de décision est simple et efficace bien qu'il ne soit pas évolutif par lui-même.

C. Réseau de neurones

Les réseaux neuronaux constituent un programme ou un modèle de machine learning qui prend des décisions d'une manière comparable au cerveau humain, en utilisant des processus qui reproduisent la façon dont les neurones biologiques fonctionnent de concert pour identifier des phénomènes, évaluer des options et arriver à des conclusions. Grâce à leur capacité de classification et de généralisation, les réseaux de neurones sont généralement utilisés dans des problèmes de nature statistique.

1. Étude des existants

Il existe plusieurs articles de chercheurs exposant un problème de jeu de poursuite/évasion utilisant l'intelligence artificielle et l'apprentissage par renforcement avec les réseaux de neurones.

L'article que nous avons choisi de présenter est : "Autonomous Decision Making for UAV Cooperative Pursuit-Evasion Game with Reinforcement Learning" rédigé par Yang Zhao, Zidong Nie, Kangsheng Dong, Qinghua Huang et Xuelong Li. Cet article s'intéresse à la prise de décision dans le cadre d'un affrontement entre deux équipes de drones dans un environnement en trois dimensions où les drones ont la capacité de communiquer et d'utiliser des armes. Les équipes peuvent être constitué d'un ou plusieurs drones mais dans le cas où il y a plusieurs drones le but est qu'ils collaborent entre eux. Dans un premier temps, l'article présente les trois méthodes les plus "traditionnelles" pour résoudre ce genre de problème :

- Théorie des jeux pour la modélisation et la résolution de scénarios de jeu de poursuite-évasion
- Théorie de l'optimisation pour modéliser le jeu de poursuite-évasion comme un problème d'optimisation décisionnelle multi-objectifs
- Décision par intelligence artificielle avec capacités d'auto-apprentissage

Malheureusement ces trois méthodes posent toutes problème. La première est trop tournée vers la recherche d'avantage à court terme et aura donc du mal à modéliser précisément un scénario sur le long-terme. La deuxième méthode ne sera pas efficace pour du temps-réel car elle prendra une décision trop lentement mais peut être utile pour l'optimisation des règles de jeu de poursuite et d'évasion. Quant à la dernière méthode, elle aura du mal à englober tous les scénarios envisageables. L'idée des chercheurs est d'utiliser le deep reinforcement learning pour permettre aux drones de prendre une décision dans un scénario de poursuite/évasion. Dans cet article plusieurs méthodes/algorithmes sont utilisés par les chercheurs mais nous avons choisi de parler uniquement celle qui a un lien direct avec cette section du document. Donc une solution envisagée par les chercheurs est : utiliser la technique heuristique Qtnetwork pour entraîner des modèles de réseaux neuronaux et permettre la décision de manœuvre dans un jeu de poursuite et d'évasion. On peut lire un peu plus loin dans l'article que le réseau de neurones est utilisé pour résoudre une équation donnant la valeur d'une action. Nous pouvons donc nous inspirer de cette méthode dans notre projet pour utiliser un réseau de neurones pour calculer les probabilités utiles à nos deux agents dans notre simulation.

2. Étude technique

Pour présenter un réseau de neurones on peut vulgariser et dire qu'un réseau de neurones est seulement une fonction à qui ont passé des paramètres et qui nous renvoie un ou plusieurs résultats. Mais il y a quand même une différence avec une fonction "normale", c'est que les réseaux de neurones ont énormément de paramètres. Cela peut même se compter en millions. Comme on peut le voir sur l'image d'illustration.

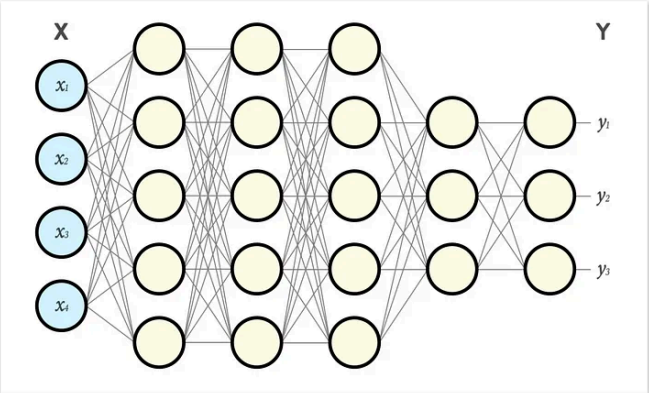


Schéma de réseaux de neurones

Un réseau de neurones est en général composé d'une succession de couches dont chacune prend ses entrées sur les sorties de la précédente. Si on regarde d'un peu plus près le fonctionnement d'un neurone, il faut savoir que chaque neurone d'une couche va recevoir en entrée toutes les valeurs que chaque neurone de la couche précédente renvoie. Et chaque valeur va être multiplié par un poids qui représente le poids de la connexion entre le neurone qui reçoit et celui qui renvoie. Toutes ces valeurs qui ont étés multiplié par un poids vont être additionnées entre elles pour devenir l'entrée du neurone qui va pouvoir la modifier en appliquant une fonction mathématique.

Cette fonction mathématique est appelée la **fonction d'activation** et elle est là pour déterminer si la valeur pourra passer aux prochains neurones. Si la valeur passe le résultat sera proche de 1, si la valeur ne passe pas elle sera proche de 0. Mais il faut aussi savoir que le neurone peut ajouter à son entrée ce que l'on appelle le biais qui est une valeur qui lui permet d'avoir une sorte d'influence sur son activation.

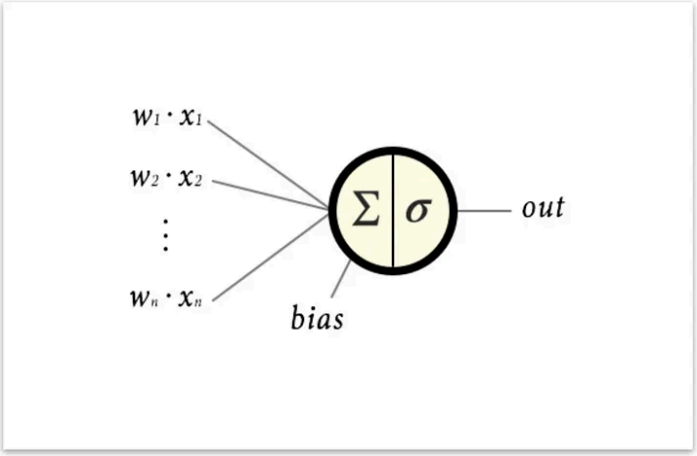


Schéma de la composition d'un neurone

Maintenant un réseau de neurones bayésien est un réseau de neurones où les poids sont des distributions probabilistes. Le réseau de neurones bayésien va être plus à l'aise pour quantifier l'incertitude et plus robuste pour de nouvelles données.

D. Apprentissage par renforcement

L'apprentissage par renforcement (ou RL pour *reinforcement learning*) est une branche de l'apprentissage automatique. La principale qualité de cet apprentissage est qu'il a une forte similarité avec les comportements humains, en matière d'apprentissage par expérience (essais-erreurs).

1. Étude des existants

L'apprentissage par renforcement est aujourd'hui utilisé dans divers projets de domaines variés tel que dans le secteur:

➢ Jeu vidéo

- **Le projet Malmo par Microsoft** : Malmo est une plateforme de recherche développée pour explorer les algorithmes de RL dans des environnements virtuels, en utilisant Minecraft comme terrain de test. Les chercheurs peuvent y configurer des scénarios d'interaction pour les agents, ce qui permet de tester diverses stratégies de RL.

- **StarCraft II AI Research** : Ce projet, soutenu par Blizzard et DeepMind, utilise StarCraft II pour explorer des approches de RL avancées. L'objectif est de construire des agents capables de gérer des stratégies complexes et de prendre des décisions rapides dans un environnement hautement dynamique.

➢ Médical

- **Traitement personnalisé des patients** : Dans la santé, le RL est utilisé pour adapter les traitements médicaux en fonction des caractéristiques des patients et de leurs antécédents médicaux. Cette approche permet de tester différentes combinaisons de traitements et d'ajuster les décisions médicales de manière dynamique pour améliorer les résultats.

➢ Robotique

- **Projet Minerva** : Dans ce projet, un robot autonome a navigué dans le Smithsonian Museum à Washington, interagissant avec les visiteurs et apprenant à éviter les obstacles. Ce projet démontre la capacité du RL à gérer des environnements physiques et complexes en temps réel et à adapter ses actions en fonction des interactions humaines.

Ces projets démontrent que l'apprentissage par renforcement peut être appliqué à des environnements variés avec des éléments uniques tels que l'interaction avec les humains, la dynamique de l'environnement et les différentes incertitudes.

2. Étude technique

Techniquement, cet apprentissage met en jeu un **agent** (dans notre cas une IA) qui interagit avec un environnement connu ou non en prenant des actions. Après chaque action, l'agent reçoit une **récompense** ou une **pénalité** en fonction de l'impact de cette action sur l'environnement.

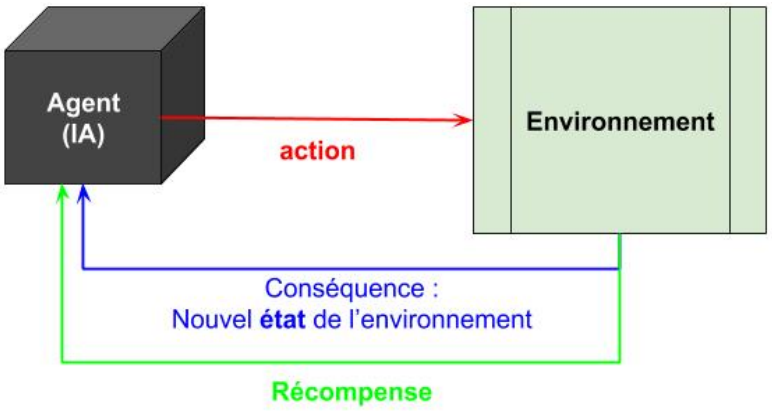


Schéma du fonctionnement classique de l'apprentissage par renforcement

L’objectif de l’apprentissage par renforcement est de maximiser la somme des récompenses accumulées sur le long terme pour prévoir les prochaines actions au mieux. Pour cela, l'agent utilise des techniques d’exploration pour tester différentes actions et en apprendre les conséquences. Au fil du temps, il développe une **politique** optimale, c’est-à-dire une stratégie qui lui permet de prendre les meilleures décisions en fonction de l'état actuel de l'environnement et des différentes données qu’il aura pu récolter.

Dans le cas de notre simulation, l’agent serait soit un gardien ou un prisonnier qui effectuerait une action donc un déplacement dans l'environnement, la carte. Ses déplacements auraient donc des conséquences sur l’environnement car l’entité se sera déplacée et aura un nouvel emplacement sur la carte.
Il existe donc **trois principes** pour l’apprentissage par renforcement classique:

- **Exploration et exploitation** : L'agent doit trouver un équilibre entre **explorer** de nouveaux endroits de la carte pour découvrir leurs récompenses potentielles et **exploiter** les endroits déjà connus pour maximiser ses gains.
- **Modèle Markovien** : L’apprentissage par renforcement s’appuie souvent sur des processus de décision Markoviens (MDP), où la prochaine étape dépend uniquement de l'état actuel et de l'action choisie, ce qui simplifie la modélisation de l'environnement.
- **Algorithmes** : Parmi les techniques d’apprentissage par renforcement, on retrouve des algorithmes comme le **Q-learning**, qui apprend une fonction de valeur pour chaque paire état-action, et les réseaux de neurones profonds utilisés dans des méthodes comme **Deep Q-Network (DQN)** pour traiter des environnements complexes, tels que ceux des jeux vidéo en 3D. (comme le projet StarCraft II AI Research)

Dans notre projet, nous utiliserons l’apprentissage par renforcement bayésien (BRL) qui introduit une dimension plus probabiliste pour gérer les incertitudes. Contrairement au RL classique, qui utilise une fonction de valeur fixe, le BRL considère que les paramètres de l’environnement sont incertains et les représente par des distributions de probabilité.

L’inférence bayésienne permet ainsi à l’agent de constamment mettre à jour ses croyances et d’adapter ses actions en conséquence, même dans des environnements partiellement observables.

L’utilisation de l’apprentissage par renforcement bayésien sera très intéressante dans ce projet pour gérer les scénarios car elle permettra de modéliser les incertitudes liées aux déplacements des autres entités et aux éventuelles zones non observables dans la carte.

E. Apprentissage génétique

Appartenant à la famille des algorithmes évolutionnistes, les algorithmes génétiques s’inspirent des principes de la génétique formelle, en utilisant le concept de transmission et de mutation de segments de « code génétique » pour optimiser progressivement une solution. L’objectif est de faire évoluer une population initiale de solutions candidates vers une solution idéale en appliquant des opérations de sélection, de croisement, et de mutation de manière itérative. À chaque génération, les solutions les plus prometteuses sont sélectionnées (fitness), combinées et mutées pour donner naissance à une nouvelle génération de solutions, augmentant ainsi les chances d’atteindre un optimum global.

1. Étude des existants

Plusieurs recherches utilisent des algorithmes génétiques pour résoudre des problèmes d'optimisation complexes. Nous nous basons ici sur l’article intitulé **"Algorithmes génétiques appliqués à la gestion du trafic aérien"**. Cet article examine l'optimisation de trajectoires pour des avions en conditions réelles, avec des contraintes de conflit et de sécurité. Les solutions proposées par les algorithmes génétiques permettent de générer des trajectoires sûres en intégrant des variables telles que le trafic environnant et les contraintes spatiales.

Dans cet article, trois méthodes principales pour la résolution de ce type de problème sont comparées :

- **Méthodes déterministes** : utiles pour des solutions rigides et contrôlées, mais limitées par un espace de recherche réduit et peu flexible.
- **Méthodes heuristiques** : capables de traiter un plus large espace de solutions mais qui peuvent manquer de précision dans la recherche de l'optimum.
- **Algorithmes génétiques** : ces méthodes sont plus adaptées à des environnements dynamiques grâce à leur capacité à explorer un espace de solutions plus large et diversifié.

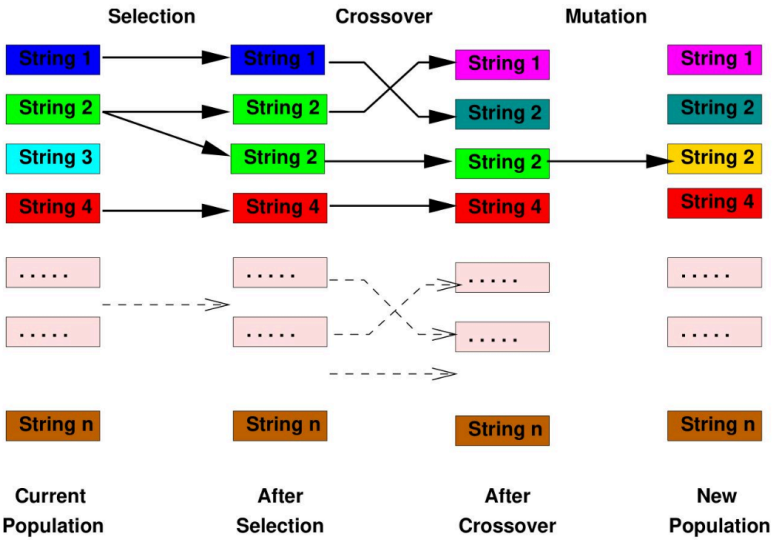
Les chercheurs suggèrent l'utilisation des algorithmes génétiques pour surmonter les limites des deux premières méthodes, en offrant une approche adaptative capable de s'ajuster aux conditions changeantes du trafic aérien. Les algorithmes génétiques permettent d'identifier progressivement des solutions optimales en fonction de contraintes multiples et de produire des résultats itératifs en temps réel.

Les éléments de la population représentent des trajectoires potentielles pour chaque avion, et les fonctions de fitness sont définies pour minimiser les conflits et optimiser la sécurité et la consommation de carburant. Des opérations de croisement et de mutation sont appliquées à chaque génération pour diversifier les solutions.

2. Étude technique

Comme expliqué précédemment, les algorithmes génétiques utilisent la notion de sélection naturelle. Les éléments nécessaires à leurs implémentations sont les suivants :

- **Codage des éléments de la population** : La première étape consiste à définir une représentation de chaque élément de la population en une solution potentielle du problème à optimiser. Cette structure permet de représenter les paramètres de chaque solution en gènes, qui seront ensuite manipulés au cours des étapes suivantes.
- **Sélection** : Une fois les solutions de la population initiale définies, un critère de sélection, aussi appelé fitness, est appliqué pour évaluer la qualité de chaque solution. Les solutions présentant les gènes les plus proches de l'objectif visé, sont sélectionnées.
- **Croisement** : Après la sélection, les solutions retenues sont croisées. Le croisement consiste à combiner certaines parties des solutions sélectionnées pour créer de nouvelles solutions. Cette opération simule le mélange génétique en héritant des caractéristiques de plusieurs « parents » pour générer des « enfants » potentiellement meilleurs. Les croisements successifs entre les meilleures solutions contribuent à la recherche d'une combinaison optimale des paramètres.
- **Mutation** : Afin d'éviter le risque de convergence prématurée vers des solutions sous-optimales, une étape de mutation est introduite. Elle consiste à modifier aléatoirement certains paramètres de solutions sélectionnées, créant ainsi une diversité dans les gènes de la population. Cette diversité est essentielle pour explorer de nouvelles régions de l'espace de solutions et maintenir un potentiel d'adaptation.
- **Nouvelle génération** : La nouvelle génération de solutions est ainsi formée, intégrant les croisements et mutations des solutions précédentes. Le processus itératif se poursuit, appliquant de nouveau les critères de sélection, de croisement et de mutation à chaque nouvelle génération jusqu'à atteindre un niveau de performance optimal ou un nombre de générations prédéfini.



Représentation d'un cycle de sélection, croisement et mutation pour un GA

Dans notre projet, l'algorithme génétique pourra être utilisé pour définir les stratégies de déplacement adaptatif les plus optimaux. Chaque solution correspond à un ensemble de comportements face à plusieurs situations et les itérations successives pourrons croiser et modifier les comportements pour trouver les plus optimaux pour nos agents.

III. Conclusion

En conclusion, ce projet de simulation poursuite-évasion entre agents informatiques intelligents peut explorer divers apprentissages automatiques, notamment l'apprentissage par renforcement bayésien. Cet apprentissage nous permettra de simuler des interactions réalistes entre gardiens et prisonniers dans un environnement donné.

La prochaine étape consistera à faire une étude préalable pour pouvoir évaluer les différents scénarios possibles et affiner nos choix techniques et les modifier si besoin.

BIbliographique

[Evasion - poursuite dans un environnement complexe (marin)]

Pursuit-evasion game strategy of USV based on deep reinforcement learning in complex multi-obstacle environment - ScienceDirect

[Robust Policies for a Multiple Pursuer Single Evader Differential Game]

mp1e-robust.pdf

- Arbre de comportement :

https://www.javacodegeeks.com/2014/08/game-ai-an-introduction-to-behaviour-trees.html

[Game AI – An Introduction to Behaviour Trees]

https://asana.com/fr/resources/decision-tree-analysis [Qu'est-ce qu'un arbre de décision? Avantages et inconvénients]

- Moteur de jeu:

https://members.loria.fr/VThomas/mediation/ISN_moteur_2017/code_java/Documentation_moteur_Jeu.pdf [Création de moteur de jeu]

https://members.loria.fr/vthomas/mediation/JV_IUT_2016/ [Création d'un jeu en java]

https://flylib.com/books/en/2.124.1.2/1/ [Practical Java Game Programming]

https://medium.com/@dsaragih/build-a-snake-ai-with-java-and-libgdx-part-1-b203d575a0cf

https://libgdx.com/[LibGDX]

https://gdevelop.io/fr-fr/game-makers [GDevelop]

https://docs.unity3d.com/Manual/index.html [Unity]

https://www.commentcoder.com/moteur-jeu/ [Classement des meilleurs moteurs de jeu]

https://www.cocos.com/en/cocos2d-x [Cocos2D]

https://peerdh.com/fr/blogs/programming-insights/top-10-java-libraries-for-building-simple-games-3 [Top 10 bibliothèques Java pour créer des jeux simples]

- ➢ Inférence bayésienne

https://www.cyberjustice.ca/2020/12/16/les-techniques-algorithmiques-de-lia-linference-bayesienne/ [Definition et explication de l'inférence bayésienne utilisée avec de l'IA]

- ➢ Inférence bayésienne - Algorithme de renforcement

https://orbi.uliege.be/bitstream/2268/166188/1/JFPDA14-v3.pdf

https://www.collectionscanada.gc.ca/obj/thesescanada/vol2/QQLA/TC-QQLA-27809.pdf [Mémoire expliquant l'apprentissage par renforcement bayésien]

https://www.microsoft.com/en-us/research/project/project-malmo/ [Projet Malmo utilisant l'inférence bayésien]

https://members.loria.fr/ADutech/Papier/pres_POMDPandMath.pdf [Algorithme POMDP]
https://www.ri.cmu.edu/project/minerva/ [Projet Minerva]

https://www.actuia.com/actualite/lapprentissage-supervise-et-par-renforcement-pour-aider-les-robots-a-sadapter-a-tout-type-denvironnement/ [Utilisation du renforcement sur de vrais robots pour s'adapter à l'environnement]

https://vitalflux.com/bayesian-machine-learning-applications-examples/#Bayesian_Machine_Learning_Applications [Exemples d'application de l'apprentissage par renforcement bayésien]

https://members.loria.fr/OBuffet/papiers/jfpda12-b.pdf [L'apprentissage par renforcement bayésien et leurs differentes approches]

https://deepmind.google/discover/blog/alphastar-grandmaster-level-in-starcraft-ii-using-multi-agent-reinforcement-learning/ [Projet StarCraft II AI Research]

- ➢ Réseaux de neurones

https://towardsdatascience.com/part-1-a-neural-network-from-scratch-foundation-e2d119df0f40

https://arxiv.org/pdf/2411.02983

https://fr.wikipedia.org/wiki/R%C3%A9seau_de_neurones_artificiels.

➢ Algorithme génétiques

http://bruno.mascret.fr/ia/algo_gen/index.html

https://www.scribbledata.io/blog/how-genetic-algorithms-are-shaping-ai-and-ml/

https://enac.hal.science/hal-00991623/document