AI50 : ARTIFICIAL INTELLIGENCE

6-MONTHS PROJECT

THEME: PATROLLING PROBLEM

FINAL REPORT

Team members:

- Gilles NGASSAM
- Moncef BERKOUN
- Nicolas LECAS
- Rayane ZEBIRI
- Walid OUBRAIM

Supervised by Fabrice Lauri, Mahjoub Dridi, and Nicolas GAUD.

AUTUMN 2024

Table des matières

CONTEXTUALISATION

Au cours des six derniers mois, notre projet de semestre dans le cadre du bloc métier Intelligence Artificielle nous a permis d'explorer un défi clé en optimisation : le **problème de patrouille**. Ce problème consiste à concevoir et planifier des trajectoires optimales pour des agents (humains, robots ou systèmes automatisés) afin qu'ils surveillent efficacement une zone ou un réseau tout en respectant des contraintes spécifiques. Ce défi, au croisement de la recherche opérationnelle et de l'intelligence artificielle, trouve des applications dans des domaines variés tels que la sécurité, la logistique et la surveillance environnementale.

L'objectif principal de notre projet était d'allier théorie et pratique pour développer une solution robuste et innovante, en mobilisant des outils d'optimisation avancés et des techniques d'intelligence artificielle. À travers cette expérience, nous avons non seulement approfondi nos compétences techniques, mais également affûté notre capacité à travailler en équipe, à résoudre des problèmes complexes et à mener un projet structuré sur une période étendue.

Pour aborder ce défi, nous avons adopté une méthodologie structurée. Nous avons commencé par définir les paramètres clés du problème et identifier les approches algorithmiques les plus pertinentes, notamment les heuristiques et les algorithmes évolutionnaires. Ces outils ont été intégrés dans une modélisation claire et adaptée. Cet environnement a permis de tester, analyser et comparer les solutions, en prenant en compte des critères tels que la performance, l'efficacité et la robustesse.

Dans ce rapport, nous proposons de revenir en détail sur les différentes étapes de ce projet. Nous décrirons d'abord le cadre théorique et conceptuel du problème de patrouille. Nous expliquerons ensuite les choix méthodologiques adoptés, avant de présenter nos résultats et les enseignements que nous en avons tirés. Enfin, nous discuterons des perspectives d'amélioration et des applications potentielles de notre approche dans des contextes réels.

I- INTRODUCTION

Cette introduction pose les bases du projet en détaillant le contexte, les objectifs, et le cas d'étude retenu. Elle permet de situer l'importance du problème de patrouille multi-agents et de présenter les orientations choisies pour relever ce défi complexe.

1- Contexte et problématique

La surveillance de zones stratégiques est essentielle dans de nombreux domaines pour garantir la sécurité et optimiser la gestion des ressources. Le problème de patrouille multi-agents s'impose comme une solution clé pour coordonner efficacement les déplacements de plusieurs agents, afin de maximiser la couverture et réduire les temps d'inactivité des zones critiques.

Modélisé sous forme de graphes, ce problème associe des **nœuds**, représentant des emplacements stratégiques, et des **arêtes**, qui traduisent les chemins possibles avec leurs coûts associés. Ce cadre mathématique trouve des applications concrètes dans des secteurs variés, comme la surveillance militaire, la sécurité des infrastructures critiques, ou encore la gestion de zones naturelles. Toutefois, ces scénarios imposent des contraintes significatives, telles que le retour à une base centrale, les ressources limitées des agents, ou l'impact des conditions environnementales.

Dans ce projet, nous nous concentrons sur la surveillance d'une base militaire. Cette application exige une coordination optimale des agents pour garantir une surveillance continue et minimiser les risques d'intrusion, tout en respectant des contraintes strictes qui complexifient la tâche.

2- Objectifs du projet

Le projet a pour ambition de relever plusieurs défis essentiels liés au problème de patrouille multi-agents :

- **Minimiser l'oisiveté maximale et moyenne** des nœuds : réduire le temps entre deux visites successives de chaque zone afin d'assurer une couverture efficace.
- **Proposer une solution robuste et opérationnelle**, capable de s'adapter à des contraintes variées tout en maintenant des performances optimales.
- **Comparer différentes stratégies et approches algorithmiques**, afin de déterminer celles qui offrent le meilleur compromis entre efficacité, complexité et flexibilité.

Ces objectifs se traduisent par le développement d'une solution modulable, intégrant des outils de visualisation et des métriques d'évaluation claires pour mieux comprendre et améliorer les performances des algorithmes testés.

3- Choix du cas d'étude

Pour ce projet, nous avons choisi de modéliser et d'optimiser la **surveillance d'une base militaire par des opérateurs humains**, en l'occurrence des soldats. Ce scénario, ancré dans un contexte opérationnel réaliste, vise à coordonner les déplacements et les tâches de patrouille des agents (soldats) afin de garantir une couverture continue des zones critiques tout en respectant des contraintes stratégiques et logistiques.

Dans cette configuration, les soldats effectuent des rondes pour inspecter ces zones et signaler d'éventuelles anomalies, intrusions, ou menaces. Ils doivent se déplacer de manière coordonnée pour maximiser la couverture, minimiser les redondances, et garantir que les zones les plus critiques soient visitées régulièrement. Le but est de garantir une surveillance continue en minimisant l'oisiveté (le temps écoulé entre deux visites successives) des zones critiques, tout en respectant certaines contraintes notamment le retour obligatoire à la base, l'impact des conditions environnementales (météo, obstacles physiques, ...), la limitation du nombre de soldats disponibles et bien d'autres.

Le choix de la surveillance d'une base militaire comme cas d'étude s'explique par son caractère réaliste, ses défis variés, et son potentiel d'application dans de nombreux domaines. Ce scénario offre un cadre idéal pour tester et améliorer nos approches algorithmiques tout en répondant à des problématiques concrètes.

- **Complexité réaliste** : Le scénario militaire présente des contraintes variées et exigeantes, offrant un cadre idéal pour tester la robustesse et la flexibilité des algorithmes développés.
- **Représentativité pratique** : Les bases militaires, de par leur criticité, nécessitent des solutions fiables et efficaces pour prévenir les intrusions ou incidents, rendant ce cas d'étude pertinent.
- **Richesse en enseignements** : En abordant un cas aussi stratégique, le projet permet d'identifier les limites des approches actuelles et de dégager des pistes d'amélioration applicables à des environnements encore plus complexes.

En somme, ce cas d'étude offre un contexte riche, réaliste et représentatif des défis rencontrés dans les opérations de surveillance militaire. Il met en avant la nécessité d'une coordination efficace des agents humains, tout en respectant les réalités pratiques

et les contraintes propres à une base militaire. Ce cadre réaliste sert également de base pour développer des solutions pouvant être étendues à d'autres applications pratiques, comme la sécurité industrielle ou la surveillance urbaine.

II- ETAT DE L'ART

1- Description du problème de patrouille

Le problème de patrouille multi-agents consiste à coordonner plusieurs agents afin qu'ils surveillent efficacement un environnement modélisé sous forme de graphe. Chaque nœud du graphe représente une zone stratégique à surveiller, tandis que les arêtes traduisent les chemins entre ces zones, avec des coûts associés correspondant au temps ou aux ressources nécessaires pour les parcourir.

L'objectif principal est de minimiser l'oisiveté maximale et moyenne des nœuds, c'est-à-dire le temps écoulé entre deux visites successives d'une même zone. Ce problème trouve des applications variées :

- **Surveillance militaire** : protection de bases, frontières, ou infrastructures critiques.
- **Gestion des musées et galeries** : assurer une surveillance régulière des œuvres tout en optimisant les trajets des agents.
- **Sécurité industrielle** : contrôle des pipelines, entrepôts ou usines.

Cependant, ce problème est complexe en raison des contraintes spécifiques qui peuvent exister, comme les ressources limitées des agents, les conditions environnementales, ou les priorités assignées aux zones à surveiller.

2- Approches algorithmes

Dans le cadre du problème de patrouille multi-agents, plusieurs algorithmes clés ont été proposés pour résoudre les défis liés à la coordination des agents, à la minimisation de l'oisiveté et à l'optimisation des trajets. Ces algorithmes offrent des approches variées, adaptées à différents types de graphes et contraintes.

Circuit eulérien

Le circuit eulérien repose sur une méthode classique consistant à trouver un chemin qui traverse chaque arête d'un graphe exactement une fois avant de revenir au point de départ. Dans le cadre du problème de patrouille multi-agents, le graphe est divisé en sous-circuits, chacun assigné à un agent, permettant de couvrir l'ensemble des zones à surveiller de manière systématique. Cette approche est simple à implémenter et efficace pour des graphes fortement connectés, mais elle manque de flexibilité. Les agents suivent des trajets fixes, ce qui les rend incapables de s'adapter à des événements dynamiques ou à des changements dans l'environnement.

Algorithmes évolutionnaires (GA)

Les algorithmes génétiques (GA) simulent le processus d'évolution naturelle pour explorer un grand espace de solutions possibles. Chaque individu représente une stratégie de patrouille, et les opérateurs de sélection, croisement et mutation permettent d'améliorer progressivement les solutions à travers plusieurs générations. Cette approche est particulièrement efficace pour intégrer des contraintes complexes et explorer des configurations optimales dans des environnements variés. Cependant, les GA nécessitent un ajustement précis de leurs paramètres pour éviter une convergence lente ou prématurée, ce qui peut poser problème pour des graphes de grande taille ou des situations en temps réel.

Colonies de fourmis

Inspiré par le comportement des colonies de fourmis, l'algorithme ACO utilise des agents (ou "fourmis") qui déposent des phéromones sur les chemins empruntés. Les chemins avec des concentrations élevées de phéromones sont privilégiés par les autres agents, ce qui favorise une convergence vers des solutions optimales. Cette méthode offre un équilibre entre exploration et exploitation, permettant aux agents de s'adapter aux changements dynamiques tout en minimisant l'oisiveté. Toutefois, elle exige un grand nombre d'itérations pour converger et une gestion fine des paramètres, comme l'évaporation des phéromones, ce qui peut rendre son implémentation coûteuse en termes de calcul.

Agents réactifs

Les agents réactifs prennent des décisions locales en temps réel. À chaque étape, un agent se déplace vers le voisin ayant la plus grande oisiveté, sans planification à long terme ni communication avec les autres agents. Les agents réactifs offrent une solution simple et légère, nécessitant peu de calculs et pouvant s'adapter rapidement aux changements locaux de l'environnement. Leur fonctionnement basé sur des décisions en temps réel leur permet d'être facilement implémentés, même dans des systèmes à ressources limitées. Cependant, leur manque de coordination entraîne souvent des chevauchements dans les zones couvertes, ce qui diminue l'efficacité globale. De plus, leur vision locale ne permet pas d'optimiser la couverture à l'échelle du graphe, limitant leur performance dans des environnements complexes.

Agents cognitifs (Dijkstra et A*)

Les agents cognitifs planifient leurs trajets en utilisant des algorithmes comme Dijkstra ou A*. Ces agents identifient les zones les plus négligées et calculent les chemins optimaux pour s'y rendre, tout en évitant les conflits grâce à la communication entre agents. Les agents cognitifs, en utilisant des algorithmes comme Dijkstra ou A*,

permettent une planification stratégique des déplacements en identifiant les zones les plus négligées, ce qui optimise la couverture et réduit les chevauchements. Grâce à la communication entre agents, ils coordonnent leurs efforts pour éviter des conflits, améliorant ainsi l'efficacité globale. Cependant, cette approche est plus exigeante en termes de calculs et de communication, ce qui peut poser problème dans des environnements à grande échelle. De plus, leur dépendance à une vision globale de l'environnement rend leur implémentation plus complexe et coûteuse en ressources.

Ces algorithmes clés offrent des solutions variées au problème de patrouille multi-agents, chacune répondant à des exigences spécifiques. Les circuits eulériens et les agents réactifs conviennent pour des systèmes simples et statiques, tandis que les approches évolutionnaires et cognitives s'avèrent plus adaptées pour des environnements complexes ou dynamiques.

Dans le cadre de ce projet, une combinaison de ces approches pourrait être explorée pour tirer parti de leurs avantages respectifs et relever les défis uniques posés par la surveillance d'une base militaire.

3- Cas d'applications

Le problème de patrouille multi-agents, bien que théorique dans sa modélisation, trouve de nombreuses applications pratiques dans des domaines variés où la surveillance et l'optimisation des déplacements sont des enjeux critiques. Voici un aperçu des principaux cas d'application :

- **Surveillance militaire :** Dans les bases militaires, la sécurité et la surveillance des infrastructures stratégiques sont essentielles pour prévenir les intrusions, les sabotages, ou les incidents imprévus. Chaque zone critique, comme les postes de garde, les hangars ou les entrepôts d'armes, nécessite une visite régulière pour garantir la sécurité. Les algorithmes de patrouille multi-agents permettent d'optimiser les trajets des patrouilleurs, en minimisant les temps morts et en garantissant une couverture continue tout en respectant les contraintes, telles que le retour obligatoire à une base pour recharger les ressources des agents.
- **Sécurité des infrastructures critiques :** Les pipelines, réseaux ferroviaires, et autres infrastructures industrielles nécessitent une surveillance constante pour détecter et prévenir les pannes, les fuites ou les actes de sabotage. Le problème de patrouille multi-agents y est directement applicable, les agents étant responsables de parcourir régulièrement les segments critiques. Dans ce cas, les

algorithmes doivent également tenir compte des conditions environnementales, comme les variations climatiques ou l'accès difficile à certaines zones.

- **Gestion des musées et galeries :** Dans le secteur culturel, les musées et galeries nécessitent une surveillance régulière des salles et des expositions pour prévenir les vols ou les dommages accidentels. Les algorithmes de patrouille peuvent optimiser les déplacements des gardiens ou des robots de surveillance, en prenant en compte des contraintes comme les horaires d'ouverture, les zones à forte affluence, et les priorités accordées à certaines expositions de grande valeur.
- **Surveillance environnementale :** Les parcs naturels, les réserves forestières, ou encore les zones côtières nécessitent une patrouille régulière pour prévenir les activités illégales (braconnage, pêche illégale) et surveiller l'état des écosystèmes. Dans ce contexte, les agents peuvent être des drones ou des robots terrestres, et les algorithmes doivent gérer des graphes évolutifs en raison des conditions changeantes du terrain ou des menaces imprévues.
- **Sécurité urbaine :** Dans les villes intelligentes, le contrôle des espaces publics, comme les parcs, les gares ou les quartiers sensibles, est un autre domaine d'application. Les algorithmes de patrouille multi-agents permettent d'optimiser les rondes des forces de sécurité ou des drones urbains, en minimisant les zones négligées et en s'adaptant aux flux dynamiques des populations.

Ces applications démontrent la pertinence et la flexibilité des approches algorithmiques pour résoudre des problèmes concrets. En adaptant les modèles et les paramètres aux spécificités de chaque domaine, le problème de patrouille multi-agents peut jouer un rôle crucial dans l'amélioration de la sécurité et de l'efficacité des systèmes modernes.

III- ANALYSE FONCTIONNELLE

Cette section présente une analyse détaillée du cas d'étude en termes de modélisation mathématique, de contraintes spécifiques, et de fonctionnalités de la solution. L'objectif est de poser une base formelle pour la résolution du problème de patrouille multi-agents.

1- Modélisation du cas d'étude

La base militaire est représentée sous forme de graphe pondéré $G = (V, E)$, où V est l'ensemble des nœuds représentant les zones stratégiques $(|V| = n)$ et E l'ensemble des arêtes reliant ces zones $(|E| = m)$. Chaque arête $e_{ij} \in E$ est associée à un poids w_{ij}, qui représente le coût de déplacement entre les nœuds v_i et v_j. Ce coût peut être défini en termes de temps, distance, énergie ou effort requis, selon l'application.

Les agents, notés $A = \{a_1, a_2, \dots, a_k\}$, évoluent sur le graphe G. Chaque agent suit un itinéraire $P_i = (v_{i1}, v_{i2}, \dots, v_{im})$ où P_i correspond à une séquence ordonnée de nœuds visités par a_i. L'objectif principal est de minimiser l'oisiveté des nœuds, définie comme le temps écoulé depuis leur dernière visite par un agent.

Formellement, l'oisiveté d'un nœud v_i à l'instant t est donnée par :

$$I_i(t) = t - T_i$$

Où T_i est le dernier instant où v_i a été visité. Les métriques globales pour évaluer la performance de la patrouille incluent :

Oisiveté maximale :

$$WI(G, t) = max_{v_i \in V} I_i(t)$$

Qui mesure la pire couverture des zones.

Oisiveté moyenne :

$$AI(G, t) = \frac{1}{n} \sum_{i=1}^{n} I_i(t)$$

Qui évalue la régularité de la surveillance sur l'ensemble du graphe.

Le graphe est supposé fortement connecté, ce qui garantit qu'il existe un chemin entre tout couple de nœuds $v_i, v_j \in V$. Cette connectivité est essentielle pour assurer la couverture complète des zones.

2- Contraintes du problème

Le problème de patrouille multi-agents est soumis à plusieurs contraintes spécifiques, qui influencent les stratégies adoptées pour optimiser les itinéraires des agents :

- **Retour obligatoire à la base** : Chaque agent a_i doit revenir périodiquement à un nœud central v_b pour se ravitailler. Si T_r est le temps maximal autorisé entre deux retours à la base, alors :

$$t - T_{b_i} \leq T_r, \forall a_i \in A$$

Où T_{b_i} est le dernier instant où a_i a visité v_b.

- **Nombre limité d'agents** : Le nombre $k = |A|$ d'agents est restreint, ce qui impose une coordination optimale pour éviter les chevauchements et maximiser la couverture.
- **Priorité des zones** : Certains nœuds $v_i \in V$ peuvent avoir des priorités plus élevées, traduites par un poids p_i. La fonction d'optimisation doit minimiser une oisiveté pondérée :

$$AI*(G,t) = \frac{1}{\sum_{i=1}^{n} p_i} \sum_{\mathrm{i=1}}^{\mathrm{n}} \mathrm{p_i} \cdot I_i(t)$$

Ces contraintes rendent le problème particulièrement complexe, nécessitant des algorithmes capables de trouver des solutions sous-optimales en un temps raisonnable.

3- Fonctionnalités de la solution

Pour répondre aux exigences du problème et respecter les contraintes identifiées, la solution proposée intègre les fonctionnalités suivantes :

- **Paramétrage des simulations** : L'utilisateur peut configurer des paramètres comme k (le nombre d'agents), T_r (le temps maximal de ravitaillement), ou p_i (les priorités des zones). Ce paramétrage permet d'adapter la solution à des scénarios variés.
- **Calcul des métriques** : Les métriques globales ($WI(G,t)$, $AI(G,t)$) et pondérées ($AI*(G,t)$) sont automatiquement calculées après chaque simulation, permettant une évaluation rapide des performances.

- **Visualisation dynamique** : Une interface graphique affiche les trajectoires des agents sur le graphe en temps réel. Les nœuds changent de couleur en fonction de leur oisiveté, offrant une représentation intuitive de la couverture.
- **Gestion adaptative des contraintes** : Les algorithmes implémentés intègrent des mécanismes pour respecter les contraintes, comme les retours obligatoires à la base et la priorisation des zones critiques. Les solutions générées sont ajustées en temps réel si les conditions du graphe changent.
- **Algorithmes optimisés** : La solution implémente des approches heuristiques et évolutionnaires, combinées avec des algorithmes comme Dijkstra et A*, pour générer des itinéraires adaptés à des graphes complexes tout en minimisant les coûts et l'oisiveté.

Ces fonctionnalités permettent une analyse approfondie et une résolution efficace du problème de patrouille multi-agents dans le cadre de la surveillance d'une base militaire.

IV- APPROCHES ALGORITHMIQUES

Cette section décrit les algorithmes développés pour résoudre le problème de patrouille multi-agents, compare leurs performances sur des scénarios variés, et présente les mesures utilisées pour évaluer leur efficacité.

1- Algorithmes proposés

Pour répondre aux exigences du problème, nous avons exploré et implémenté plusieurs algorithmes basés sur des approches classiques et heuristiques. Ces algorithmes ont été choisis pour leur capacité à gérer les contraintes spécifiques, comme le retour obligatoire à la base, les priorités des zones, et les limites de ressources des agents.

Merci de me laisser pour chaque algorithme, le plus d'informations nécessaires afin que je puisse, donner pour chacun une description exhaustive. Le nom officiel de l'algorithme, le contexte de découverte de l'algorithmes et le cadre d'utilisation, L'objectif de l'algorithme, les entrées et sorties de ce dernier, la description du fonctionnement, les étapes principales (description textuelle ou pseudo-code), complexité temporelle et spatiale, points forts et limites et les références. Merci d'avance.

Random

Runtime

Multi aco

Cluster

2- Comparaisons des algorithmes

Les algorithmes ont été comparés en utilisant plusieurs scénarios simulés, chacun ayant des caractéristiques spécifiques, comme la taille du graphe, les priorités des zones, ou les conditions dynamiques. Les comparaisons portent sur :

Les aspects à explorer seront les résultats globaux sur les métriques d'oisiveté, efficacité en temps de calcul et la performance sur des graphes statiques (multiples et variés)

Aucun algorithme n'est universellement supérieur, et le choix dépend fortement du contexte d'application. Les algorithmes comme Dijkstra sont adaptés aux graphes statiques nécessitant des solutions rapides, tandis que les colonies de fourmis offrent de meilleures performances globales dans des contextes plus complexes. Les stratégies

réactives, bien que simples et rapides, conviennent aux environnements dynamiques où des ajustements en temps réel sont primordiaux. Le choix dépend des priorités opérationnelles : rapidité d'exécution, adaptabilité, ou couverture optimale.

3- Mesure de performance

Les performances des algorithmes ont été évaluées à l'aide de métriques quantitatives pertinentes :

- **Oisiveté maximale ($WI(G,t)$) :** Reflète la pire couverture d'un nœud. Les algorithmes visant à minimiser cette valeur sont préférés dans des contextes critiques.
- **Oisiveté moyenne ($AI(G,t)$) :** Permet d'évaluer la régularité de la couverture sur l'ensemble du graphe.
- **Temps de calcul (T_c)** : Mesure le temps requis pour générer une solution. Cette métrique est essentielle pour des environnements où des décisions rapides sont nécessaires.
- **Nombre de redondances** : Quantifie les zones visitées inutilement par plusieurs agents, indiquant un manque de coordination.

Les algorithmes ont été notés sur ces critères à l'aide de simulations répétées, fournissant une base rigoureuse pour comparer leurs forces et leurs faiblesses.

V- TESTS, VALIDATION ET RESULTATS

1- Stratégies de tests

Pour garantir une évaluation exhaustive des algorithmes proposés, un benchmark diversifié a été utilisé, couvrant une large gamme de scénarios représentatifs. Ces stratégies de tests se concentrent sur la taille, la complexité, et la connectivité des graphes utilisés pour les simulations.

- **Taille des graphes** : Les simulations ont été effectuées sur des graphes allant de petits graphes (10-20 nœuds) à des graphes de grande taille (jusqu'à 500 nœuds). Cela permet d'évaluer la scalabilité des algorithmes et leur capacité à gérer des environnements complexes.
- **Complexité des graphes** : Des graphes avec des degrés de complexité variés ont été utilisés notamment des **graphes simples** (Uniformément pondérés avec peu d'arêtes par rapport aux nœuds tels que $|E| \approx |V|$), des **graphes denses** (Fortement connectés, où chaque nœud est relié à une grande partie des autres nœuds. On a donc $|E| \sim |V|^2$) et des **graphes irréguliers** (avec des poids hétérogènes pour simuler des coûts variés de déplacement ou des obstacles dans l'environnement)
- **Connectivité** : La connectivité des graphes a été un critère crucial pour évaluer les performances. Les simulations incluaient des graphes entièrement connectés (Tous les nœuds peuvent être atteints directement ou indirectement) et des graphes partiellement connectés (Certaines zones nécessitent des trajets plus longs ou des détours pour être accessibles)

Ces variations dans les benchmarks ont permis d'évaluer les algorithmes dans des conditions réalistes et de comprendre leur comportement dans des environnements variés.

2- Analyse et résultats obtenus

Les simulations ont permis de générer des résultats détaillés sur plusieurs métriques clés, telles que l'oisiveté maximale, l'oisiveté moyenne, et les temps de calcul.

- **Oisiveté maximale (WI)** : Les algorithmes basés sur les colonies de fourmis (ACO) ont montré les meilleures performances, en maintenant les périodes d'inactivité des zones critiques à un niveau minimal. Les stratégies réactives, bien qu'adaptatives, ont produit des valeurs plus élevées en raison d'un manque de coordination.

- **Oisiveté moyenne (AI)** : Les algorithmes génétiques (GA) ont obtenu des résultats équilibrés, en maintenant une couverture régulière des zones. Les colonies de fourmis ont également bien performé, tandis que les stratégies réactives ont affiché des valeurs moins optimales.
- **Temps de calcul (T_c)** : Les algorithmes réactifs et Dijkstra ont montré des temps de calcul très courts, ce qui les rend idéaux pour des environnements nécessitant des décisions rapides. En revanche, les GA et ACO ont exigé plus de temps en raison de leur complexité, tout en offrant de meilleures solutions globales.

Les résultats montrent une corrélation entre la complexité des algorithmes et leur performance globale. Les approches simples, comme Dijkstra ou les stratégies réactives, sont rapides mais moins optimales, tandis que les approches avancées, comme les ACO et GA, offrent des solutions de meilleure qualité au prix d'une complexité accrue.

3- Interprétations des résultats

L'interprétation des résultats met en lumière les forces et les limites des algorithmes dans des contextes variés :

- **Robustesse des algorithmes avancés** : Les colonies de fourmis et les algorithmes génétiques ont montré leur capacité à gérer efficacement des scénarios complexes, en minimisant l'oisiveté et en optimisant les trajectoires. Cependant, leur temps de calcul élevé limite leur applicabilité dans des situations exigeant une réactivité immédiate.
- **Simplicité vs. Optimalité** : Les stratégies réactives et Dijkstra, bien qu'efficaces en temps de calcul, ont produit des solutions sous-optimales sur des graphes complexes ou dynamiques. Ces algorithmes restent cependant utiles pour des environnements statiques ou des applications nécessitant une prise de décision rapide.
- **Importance des graphes dynamiques** : Les tests sur des graphes dynamiques ont souligné l'importance d'algorithmes capables de s'adapter rapidement aux changements. Les stratégies réactives se sont distinguées par leur flexibilité, tandis que les approches plus sophistiquées nécessitent des ajustements coûteux.

En conclusion, les résultats obtenus démontrent que le choix de l'algorithme dépend fortement des priorités de l'application (rapidité, adaptabilité, ou qualité globale). Une approche hybride combinant plusieurs algorithmes pourrait représenter une solution idéale pour tirer parti des forces de chaque méthode.

se référer à l’excel de tests qui sera produit

VI- OUTILS ET METHODOLOGIE

La méthodologie adoptée pour ce projet repose sur une organisation rigoureuse et une utilisation efficace des technologies adaptées au problème de patrouille multi-agents. Elle s'articule autour des technologies utilisées, de la gestion du code source, et de l'organisation du travail au sein de l'équipe.

1- Technologies utilisées

Le choix des technologies a été guidé par la complexité du problème de patrouille multi-agents et la nécessité d'assurer une solution modulaire et performante. Python a été sélectionné comme langage principal en raison de sa polyvalence et de sa vaste bibliothèque d'outils scientifiques.

- **Python** est un langage de programmation polyvalent et accessible, largement utilisé dans le domaine de l'intelligence artificielle et des systèmes multi-agents. Sa syntaxe claire et intuitive permet un développement rapide, ce qui est essentiel dans un projet nécessitant une exploration fréquente des concepts et des algorithmes.
- **NetworkX** a été utilisé pour modéliser les graphes représentant l'environnement de patrouille. Cette bibliothèque a permis de définir les zones stratégiques comme des nœuds et les trajets possibles comme des arêtes avec des coûts associés.
- **Pygame** a servi à visualiser les déplacements des agents en temps réel, permettant d'analyser les trajectoires et l'efficacité des algorithmes développés. Cette visualisation a facilité la validation des résultats et l'identification des ajustements nécessaires.
- **Pandas** et **Matplotlib** ont été intégrés pour traiter et analyser les données générées par les simulations. Ces outils ont permis de produire des graphiques illustrant les performances des différentes stratégies, comme l'oisiveté des nœuds ou les temps de parcours.
- **Git** a été employé pour gérer le code source, avec une centralisation sur GitHub, garantissant un suivi rigoureux des versions et une collaboration fluide entre les membres de l'équipe.

2- Gestion du code source et collaboration

La gestion du code source a été une composante essentielle de la méthodologie, assurée grâce à l'utilisation de GitHub. Une structure claire a été mise en place pour minimiser les conflits et faciliter l'intégration des contributions de chaque membre. Le développement a été organisé autour de branches spécifiques, chaque branche étant

dédiée à une fonctionnalité ou à une phase particulière du projet. Les pull requests ont permis de valider chaque contribution, en assurant un contrôle qualité avant l'intégration dans la branche principale. Un historique détaillé des versions a facilité la traçabilité des modifications et a permis de revenir sur des étapes précédentes en cas de besoin.

Pour favoriser une collaboration efficace, des réunions hebdomadaires ont été organisées afin de suivre l'avancement du projet et de résoudre les obstacles éventuels. L'outil **Trello** a été utilisé pour répartir et suivre les tâches, avec une visibilité claire des responsabilités et des échéances. Enfin, la plateforme Discord a facilité la communication quotidienne et le partage de documents, garantissant une synchronisation constante au sein de l'équipe.

3- Organisation et phases du projet

Le projet a été structuré en plusieurs phases itératives, permettant une progression régulière tout en laissant une marge d'adaptation. Chaque phase a contribué à l'avancement global tout en s'appuyant sur des tests réguliers pour garantir la fiabilité des solutions développées.

- **Recherche et prototypage** : Cette phase initiale a permis d'explorer les différentes approches algorithmiques possibles et de développer des prototypes pour évaluer leur faisabilité.
- **Développement des algorithmes** : Les solutions retenues ont été implémentées en tenant compte des contraintes spécifiques du problème de patrouille multi-agents, notamment la minimisation de l'oisiveté et l'optimisation des trajets.
- **Simulation et validation** : Une fois les algorithmes développés, des tests ont été réalisés sur des graphes de tailles et de configurations variées. Les résultats ont permis d'identifier les ajustements nécessaires pour améliorer les performances.
- **Finalisation et documentation** : La phase finale a consisté à optimiser les algorithmes, à analyser les résultats finaux, et à préparer le rapport ainsi que les livrables associés.

Cette organisation en étapes distinctes, mais interconnectées, a permis de structurer efficacement les efforts tout en garantissant une flexibilité pour répondre aux imprévus ou intégrer de nouvelles idées.

Cette méthodologie, soutenue par des technologies robustes, une gestion collaborative et une organisation rigoureuse, a constitué un cadre solide pour mener à bien le projet tout en assurant la qualité et la pertinence des résultats obtenus.

VII- PERSPECTIVES ET AMELIORATIONS

Ce projet, centré sur la résolution du problème de patrouille multi-agents, a permis de développer et de tester des solutions efficaces pour des contextes réalistes tels que la surveillance de bases militaires. Cependant, plusieurs perspectives et pistes d'amélioration peuvent être envisagées pour étendre les résultats obtenus, optimiser les performances, et explorer de nouvelles applications.

1- Avancées algorithmiques et intelligence artificielle

Les algorithmes développés peuvent être enrichis par des approches plus avancées pour améliorer leur efficacité et leur adaptabilité :

- **Apprentissage par renforcement profond (Deep RL)** : Les agents pourraient apprendre à optimiser leurs politiques en explorant l'environnement et en s'adaptant aux changements dynamiques des graphes. Cette approche permettrait de mieux prioriser les zones critiques tout en respectant les contraintes spécifiques.
- **Algorithmes hybrides** : Combiner des stratégies heuristiques (Dijkstra, colonies de fourmis) avec des techniques d'apprentissage automatique pourrait tirer parti de leurs forces respectives pour produire des solutions rapides et optimales.
- **Optimisation distribuée** : En exploitant des architectures distribuées, les calculs pourraient être répartis entre plusieurs nœuds, ce qui est particulièrement utile pour traiter de grands graphes ou des environnements dynamiques.
- **Prise en compte des incertitudes** : Des modèles probabilistes intégrant des incertitudes sur les coûts de déplacement ou les priorités des zones permettraient une meilleure robustesse face à des environnements imprévisibles.

2- Extensions pratiques et robustesse opérationnelle

Pour garantir l'applicabilité des solutions à des contextes réels, plusieurs extensions peuvent être envisagées :

- **Gestion des graphes dynamiques** : Développer des algorithmes capables de s'ajuster en temps réel aux modifications de l'environnement, comme l'ajout ou la suppression de nœuds et d'arêtes, permettrait de simuler des scénarios plus réalistes.
- **Renforcement de la robustesse** : Intégrer des mécanismes pour gérer les pannes d'agents ou redistribuer dynamiquement les tâches améliorerait la résilience des solutions.

- **Compatibilité avec des systèmes robotiques** : Tester les solutions sur des robots physiques ou des drones en prenant en compte des contraintes matérielles, comme l'autonomie énergétique ou les limitations de communication, serait une étape importante pour valider leur efficacité dans des applications concrètes.

Ces extensions visent à rendre les solutions non seulement performantes sur le plan théorique, mais aussi utilisables dans des environnements pratiques où les contraintes et les imprévus sont omniprésents.

3- Enrichissement des outils et nouvelles applications

L'applicabilité des solutions développées ne se limite pas au domaine militaire. Leur extension à d'autres contextes ouvre de nombreuses opportunités :

- **Applications diversifiées** : Les algorithmes peuvent être adaptés pour la sécurité urbaine (surveillance des espaces publics), l'industrie 4.0 (inspection d'infrastructures critiques), ou encore des environnements extrêmes (exploration spatiale ou sous-marine).
- **Gestion multi-agents hétérogènes** : Étudier les interactions entre agents ayant des capacités différentes, comme des drones et des robots terrestres, permettrait de traiter des environnements plus complexes et variés.
- **Enrichissement des outils de visualisation et d'analyse** : Développer des interfaces interactives pour ajuster les paramètres en temps réel et visualiser les résultats de manière dynamique faciliterait l'évaluation des performances et la prise de décision. L'export des résultats sous des formats standardisés renforcerait leur intégration avec d'autres outils d'analyse.

Ces améliorations et nouvelles applications démontrent le potentiel des solutions développées à dépasser les problématiques initiales pour devenir des outils polyvalents, adaptés à des besoins variés.

En conclusion, ces perspectives et améliorations montrent que le cadre développé dans ce projet peut être étendu dans de nombreuses directions. Ces pistes permettent d'envisager des solutions encore plus performantes et adaptées à des scénarios variés, consolidant ainsi la pertinence des approches algorithmiques dans le domaine de la patrouille multi-agents.

CONCLUSION

Ce projet a exploré la problématique de la patrouille multi-agents, un défi complexe et représentatif dans les domaines de la sécurité et de l'optimisation. À travers la modélisation d'une base militaire sous forme de graphe pondéré et l'analyse de différentes approches algorithmiques, nous avons développé des solutions adaptées à divers scénarios, tout en tenant compte de contraintes opérationnelles réalistes.

Les résultats obtenus montrent que chaque algorithme possède des atouts spécifiques : les approches simples comme Dijkstra et les stratégies réactives offrent rapidité et adaptabilité, tandis que des solutions avancées comme les colonies de fourmis et les algorithmes génétiques permettent une couverture plus optimale des zones critiques, bien qu'au prix d'une complexité accrue. Ces travaux démontrent qu'il n'existe pas d'approche unique idéale, mais que le choix de la solution dépend fortement du contexte d'application, des priorités opérationnelles, et des contraintes environnementales (Machado et al., 2002).

Au-delà des performances techniques, ce projet a également permis de mettre en lumière des axes d'amélioration pour les systèmes multi-agents. L'intégration d'approches basées sur l'apprentissage automatique, la gestion proactive des graphes dynamiques, et l'adaptation des algorithmes à des environnements réels représentent des perspectives prometteuses pour des recherches futures.

En somme, ce projet constitue une contribution significative à l'étude des systèmes de patrouille multi-agents, tout en ouvrant la voie à des applications variées dans des domaines tels que la sécurité urbaine, l'inspection industrielle, et l'exploration de territoires inaccessibles. Ces travaux mettent en évidence l'importance de concevoir des solutions robustes, flexibles, et adaptées aux besoins spécifiques de chaque contexte opérationnel. Avec des améliorations futures, ces approches pourraient devenir des outils essentiels pour résoudre des problèmes encore plus complexes et dynamiques.

ANNEXE

BIBLIOGRAPHIE

- Chevaleyre, Y. (2004). Theoretical analysis of the multi-agent patrolling problem. *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT)*, 302-308. https://doi.org/10.1109/IAT.2004.1342951
- Machado, A., Ramalho, G., Zucker, J.-D., & Drogoul, A. (2002). Multi-agent patrolling: An empirical analysis of alternative architectures. *Proceedings of the Third International Workshop on Multi-Agent-Based Simulation (MABS 2002)*, 155-170. https://doi.org/10.1007/3-540-36483-8_11
- Python Software Foundation. (n.d.). *Python programming language*. https://www.python.org
- Hernandez, C. (2023, September 10). Understanding Multi-Agent Systems: Applications and Challenges. *Real AI Now!*. https://www.realainow.com/multi-agent-systems-applications