Modélisation et Analyse des comportements d'une société d'agents mobiles à l'aide des réseaux de Petri

Adel Djellal

وزارة التعليم العالي والبحث العلمي

BADJI MOKHTAR – ANNABA UNIVERSITY
UNIVERSITE BADJI MOKHTAR – ANNABA

جامعة باجي مختار– عنابة

Faculté : Sciences de l'Ingénieur **Année : 2016**
Département : Electronique

THESE

Présentée en vue de l'obtention du diplôme de :

DOCTORAT en Sciences

Intitulée

Modélisation et Analyse des comportements d'une société d'agents mobiles à l'aide des réseaux de Petri

Option : Automatique

Par : DJELLAL Adel

DIRECTEUR DE THESE : LAKEL Rabah Professeur Université Annaba

Soutenance publique le 04 Octobre 2016

DEVANT Le JURY

PRESIDENT :	**ABBASSI Hadj Ahmed**	**Professeur**	**Université Annaba**
EXAMINATEURS :	**FEZARI Mohamed**	**Professeur**	**Université Annaba**
	MOUSSAOUI Abdelkrim	**Professeur**	**Université Guelma**
	BOUKROUCHE Abdelhani	**Professeur**	**Université Guelma**
	MEHENNAOUI Mohamed L.	**M.C.-A**	**Université Skikda**
INVITE :	**BOULEBTATECHE Brahim**	**M.C.-B**	**Université Annaba**

Résumé (Arabe)

تقدم هذه الأطروحة تطوير والتحقق النظري والعملي من تقنيات البحث التي نفذتها مجموعة من الروبوتات للبحث عن الهاربين في محيطات مختلفة. تم تنفيذ نموذج طوبوغرافي للمحيط على شكل مخططات للمناطق المتصلة. تم التحقق من صحة نتائج التقنيات المقترحة، وتخفيض عدد الروبوتات المتحركة اللازمة لنجاح عملية البحث، عن طريق استخدام شبكات بيتري. لحل مشكلة الملاحة بين المناطق، قد اقترحت تقنيات تعتمد على الحقول المحتملة لتوليد مسارات آمنة (للابتعاد عن العقبات) وسريعة للوصول إلى الهدف. تم إجراء المصادقة على التقنيات المقترحة على منصة المحاكاة *MobotSim*، على الأوساط الحقيقية والمصممة من جهة، والممارسة على اثنين من الروبوتات *POB-BOT* من جهة أخرى.

كلمات البحث: مجموعة الروبوتات، السعي-التهرب، الملاحة، الروبوت المتحرك، شبكات بيتري، نظرية المخططات.

Abstract

This thesis presents theoretical and implemented design and validation of search techniques, done by robot society, in pursue of evaders in various environments. The environment was topologically modeled by graphs of connected areas. The results of the proposed techniques, and the optimization of needed number for clearing operation, were validated using Petri nets. For the robot navigation problem between different areas, navigation techniques based on potential field propagation were anticipated to generate trajectories, safe in one hand (distant from obstacles) and fast in the other hand, to reach target. The validation of the proposed techniques was done using simulation platform MobotSim on designed and real environments in one hand, and a real hardware platform composed of two POB-Bot robots and adapted environments in the other hand.

Keywords : robot society, pursuit-evasion, mobile robot, navigation, Petri net, graph theory.

Résumé

Cette thèse présente l'élaboration et la validation théorique et pratique des techniques de recherche, menés par des robots organisés en société, à la poursuite d'évadés dans des environnements variés. Une modélisation topologique de l'environnement en graphes de zones connexes a été réalisée. Les résultats des techniques proposées, et l'optimisation du nombre de robots mobiles nécessaire pour la réussite de l'opération de décontamination, ont été validés par l'utilisation les réseaux de Petri. Pour le problème de navigation du robot entre les zones, des techniques basées sur la propagation des potentiels ont été proposés pour générer des trajectoires à la fois sûres (éloignées des obstacles) et rapides pour atteindre la cible. La validation des techniques proposées a été réalisée en simulation sur la plate-forme logicielle MobotSim, sur des environnements conçus et réels d'une part, et en pratique sur la plate-forme matérielle composée de deux robots POB-Bot et d'environnements adaptés d'autre part.

Mots-clés : société de robots, poursuite-évasion, robot mobile, navigation, réseau de Petri, théorie des graphes

A mon père (Repose en paix), ma mère et ma femme, et à celui qui est en train de lire ça...

Remerciements

Mes premiers remerciements vont à *Allah* tout puissant pour le courage, la patience et la volonté qu'il m'a donnée pour accomplir ce travail.

Je tiens à exprimer mes sincères remerciements et ma gratitude à mon encadreur Docteur *Rabah Lakel*, Professeur au département d'électronique de l'université d'Annaba, d'avoir accepté la direction de ma thèse, pour ses conseils, sa disponibilité et sa confiance.

J'adresse également mes remerciements à l'ensemble des membres de jury :

Professeur *Abassi Hadj Ahmed*, de l'université d'Annaba, d'avoir accepté de présider ce jury.

Professeur *Fezzari Mohamed* et Docteur *Boulebtateche Brahim*, de l'université d'Annaba, Professeur *Moussaoui Abdelkrim* et Professeur *Boukrouche Abdelhani* de l'université de Guelma, et Docteur *Mehannaoui Lamine*, de l'université de Skikda, d'avoir accepté d'examiner ce travail.

Je tiens aussi à remercier du fond de mon cœur mes parents, ma femme, mon frère et mes sœurs qui m'ont beaucoup encouragé durant ma thèse. Merci pour votre patience et vos encouragements.

Je remercie mes amis, *Bellazoug Ali*, *Hammoud Mohamed Salah* et *Lakehal Brahim* pour leurs encouragements et leurs conseils.

Enfin, merci à tous ceux qui m'ont aidé de proche ou de loin.

Table des matières

Table des figures

Liste des tableaux

Liste des Algorithmes

Introduction Générale

L'une des principales motivations du développement de la robotique mobile demeure la substitution de l'être humain en milieu hostile. Les robots sont conçus pour opérer dans différentes conditions, en prenant en compte des missions à périodes de temps plus long sans intervention humaine, Il s'agit notamment de tâches spécifiques dans un environnement dangereux ou hostile tels que les milieux industriels mais pas uniquement, car actuellement existe une forte tendance à élargir les milieux où évoluent les robots, organisés en société, à des environnements culturels (musées), commerciaux (banques, centres d'affaires, etc.).

Parmi les tâches spécifiques que peut réaliser un robot, ou mieux une société de robots, on retient principalement celles de sécurisation des lieux sensibles et de localisation des blessés bloqués dans des bâtiments partiellement détruits suite à une catastrophe naturelle. La tâche de sécurisation des lieux sensibles, présentée dans la littérature sous plusieurs appellations : "problème de poursuite-évasion", "problème policier-voleur", consiste à décontaminer l'environnement en localisant les intrus et à les neutraliser.

Il est évident que l'exploration intelligente de l'espace de recherche permet de minimiser le temps et le nombre de robots mobilisés.

La performance du système robotique est fortement dépendante de son degré d'autonomie ; C'est un axe de recherche où se concentrent beaucoup d'efforts pour améliorer les problèmes de localisation, de planification de trajectoire et de navigation.

Il est important de noter que les tâches de planification de trajectoire et de navigation sont tributaires de la bonne exécution de la tâche de localisation. En effet la localisation est l'une des fonctions essentielles qui permet aux robots mobiles de se déplacer en respectant les règles élémentaires de sécurité et d'évoluer, de façon générale, vers une autonomie totale. C'est pourquoi la recherche dans le domaine de la robotique privilégie la voie de la localisation du robot dans son environnement d'évolution. Elle est basée sur le principe de calcul et d'actualisation de la connaissance de la position et de l'orientation du robot dans un repère absolu lié à l'environnement d'une part et à la connaissance de sa position relative par rapport aux autres robots de sa communauté.

Il faut noter aussi que la performance du système de localisation final en terme de temps de calcul et de précision dépend de la performance de chacune des étapes de calcul en ligne mais aussi du choix effectué quant au modèle des capteurs et à celui de l'en-

vironnement. En plus actuellement la localisation des robots mobiles est un thème de recherche ouvert, puisque aucune méthode globale n'est susceptible de générer des algorithmes suffisamment robustes, rapides et fiables pour être appliqués à tout type de problèmes.

Les agents mobiles retenus dans notre travail sont soit des agents physiques (robots réels) ou bien des agents informatiques (programmes simulant le comportement des robots). Les environnements où évoluent les agents mobiles, aussi bien réels que simulés, représentent sous forme stylisée les musées, les banques les centres d'affaires etc.

L'infrastructure développée, pour valider les résultats de la simulation réalisée sur la plate forme logicielle MobotSim, comprend une communauté d'agents robots composée de deux robots de types POB-Bot , deux environnements tests, un protocole de communication centralisée permettant l'échange des messages concernant leurs positions relatives d'une part et l'évolution de la tâche à réaliser d'autre part et une caméra pour la localisation des robots en temps réel. Le travail expérimental a été assuré par des équipements fournis par le Laboratoire d'Automatique et Signaux de Annaba (LASA).

Afin d'effectuer la synthèse d'une commande robuste, il convient d'établir un modèle de spécifications du système considéré afin de dégager toutes les contraintes liées à la réussite de l'opération de décontamination et à la sécurisation des déplacements des robots vers leurs points dans l'espace de recherche.

A cette fin, notre choix s'est porté sur l'utilisation de la l'outil des graphes pour modéliser les environnements d'une part et de l'utilisation de l'outil des réseaux de Pétri pour valider le comportement des agents d'autre part. La théorie des graphes et les algorithmes d'optimisation qui lui sont associés permettent de décomposer l'espace en régions et de déterminer de manière optimale le nombre d'agents nécessaires pour la réussite de la mission.

Les réseaux de Petri représentent un formalisme puissant et reconnu pour la modélisation et l'analyse des systèmes à événements discrets, et les nombreuses extensions des réseaux de Petri qui ont été proposées dans la littérature, permettent de prendre en charge les différentes fonctionnalités du système ainsi que ses caractéristiques temporelles de manière explicite.

Le manuscrit est organisé comme suit :

— Dans un premier temps, nous donnons quelques généralités concernant la robotique mobile en général ainsi que tous les points devant être abordés pour une meilleure compréhension de la navigation en robotique mobile. Ce premier chapitre commence par une étude des robots mobiles, de leur architecture et des différents concepts nécessaires au problème de la navigation.
— Nous présentons ensuite, dans le deuxième chapitre, une explication des techniques proposées pour résoudre le problème de Poursuite Évasion dans un environnement complexe. On commence par réaliser une modélisation topologique de l'environ-

nement en graphes de zones connexes et on valide les résultats des techniques proposées par des réseaux de Petri.

— Dans le troisième chapitre on présentera les méthodes, basées sur les techniques de propagation des potentiels, pour générer des trajectoires à la fois sures (éloignées des obstacles) et rapides pour atteindre la cible.

— Dans le quatrième et dernier chapitre, on présente les résultats obtenus sur l'environnement de simulation MobotSim d'une part, et sur la plate-forme matérielle développée d'autre part.

Chapitre 1

État de l'art

1.1 Introduction

Ce chapitre est consacré à la présentation de l'état de la recherche en matière de robotique mobile autonome, notamment les robots organisés en société, d'une part, et à la présentation des puissants outils graphiques tel que la théorie des graphes pour modéliser l'espace de travail des robots et les réseaux de Petri pour modéliser l'évolution dynamique de la mission de sécurisation d'un environnement donné.

Ce chapitre traite aussi la détermination du nombre optimal de robots nécessaires à l'accomplissement de la mission de sécurisation d'une environnement, préalablement en zones. Les machines à états finis (FSM) sont utilisées durant la phase de planification de trajectoires et permettent d'optimiser les trajectoires obtenues en satisfaisant deux importants critères à savoir la sécurité des déplacements et le temps de parcours.

On commence par présenter d'abord la robotique mobile autonome et les besoins induits en matière de modélisation de l'espace de travail, de localisation d'un robot par rapport à son environnement et par rapport aux autres robots au sein de la société, et ensuite l'organisation de plusieurs robots mobiles sous forme d'une société, et la gestion des difficultés liées au partage des ressources communes (déplacement des robots sur des tronçons de trajectoire commune : problème d'évitement des collisions) et à la satisfaction de l'objectif commun, qui est la réalisation de la mission finale (problème de coopération).

La théorie des graphes est utilisée pour décomposer l'espace de travail en zones ou régions et permet d'optimiser le nombre d'agents robots nécessaires à la décontamination d'un bâtiment dans le cas de recherche d'intrus.

L'outil Réseaux de Pétri (RdP) permet de modéliser l'opération de décontamination de l'environnement et de valider le nombre de robots retenus pour sa réalisation d'une part, et de gérer le protocole d'échanges de messages entre les agents d'autre part.

1.2 La robotique mobile autonome

Le terme robot est apparu pour la première fois pour désigner un serviteur esclave ou un travail forcé, et c'est l'écrivain tchèque Karel Capek qui l'a utilisé pour les besoins d'une pièce de théâtre. Et celui qui est considéré comme étant le père de la robotique moderne, Joseph Engelberger, a déclaré qu'il était incapable de définir un robot, par contre il est en mesure de le reconnaître dès qu'il le voit.

Les robots mobiles autonomes, contrairement à la majorité des robots industriels, confinés à évoluer dans un espace de travail spécifique, peuvent se déplacer d'un endroit à un autre sans l'aide extérieure d'un opérateur humain. Et cette mobilité les rend aptes pour une large gamme d'applications dans des environnements divers. En fonction de la nature de l'espace de travail et du mode de locomotion, les robots mobiles se déclinent en plusieurs catégories. Les véhicules autonomes aériens [1] et les véhicules autonomes sous-marins [2] sont intimement liés à l'environnement dans lequel ils évoluent. Par contre pour les applications terrestres les robots mobiles à roues [3] sont très populaires devant les robots mobiles à pattes [4] réservés à quelques environnements dans lesquels se trouvent des zones escarpés ou contenant des marches d'escalier.

Les questions relatives à la complexité mécanique, la consommation d'énergie, et la nature du robot (holonome, omnidirectionnel), sont étudiées dans le chapitre réservé à la partie pratique de ce travail de recherche.

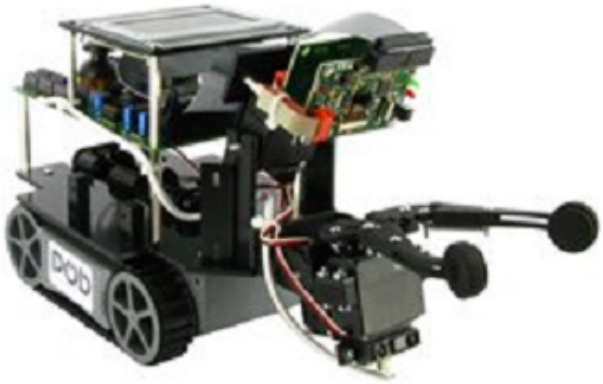


FIGURE 1.1 – Robot POB-Bot utilisé dans l'implémentation matérielle

1.3 Problème de localisation

Le problème de localisation est un problème majeur en robotique mobile, la tâche de navigation (déplacement du robot d'un point A vers le point B) ne peut se réaliser

1. Unmanned Aerial Vehicles ou *UAV*
2. Autonomous Underwater Vehicles ou *AUV*
3. Wheeled Mobile Robot ou *WMR*
4. Legged Mobile Robot ou *LMR*

que si le robot est en mesure de déterminer à chaque instant sa position courante. Il n'existe pas de capteurs donnant directement la position du robot, la posture du robot (position et orientation) est déterminée à partir du traitement de données de différentes natures provenant de mesures de différents capteurs et des estimations générées par des modèles. Dans le cas de la localisation locale, la détermination de la position courante du robot se fait alors de façon itérative en combinant la position précédente avec une estimation du déplacement fourni par un modèle du robot. Les défauts de cette technique de localisation sont liés au fait qu'elle donne naissance à une dérive provenant de la propagation et de l'amplification des imprécisions tout le long du processus d'itération (figure 1.2). La posture du robot est régulièrement encadrée en utilisant des connaissances à priori (utilisation de cartes) et en faisant des mesures extéroceptives (sonar, télémètre laser, etc.). La localisation globale est plus difficile, elle donne la position du robot par rapport à son environnement [28]. Le problème de localisation multi-robots peut être abordé par deux approches, la première approche consiste à localiser chaque robot de façon individuelle, alors que la deuxième approche intègre les informations fournies par les autres robots sur leurs positions relatives, du fait qu'ils communiquent entre eux. Il est évident que la deuxième approche, en fusionnant les données, permet d'améliorer la robustesse du processus de localisation [28].

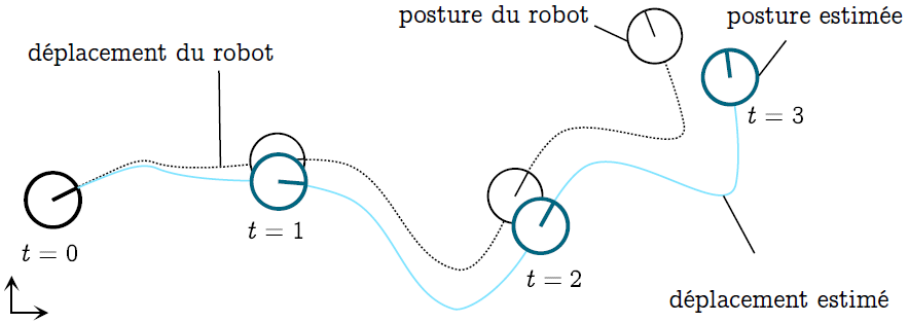


FIGURE 1.2 – Illustration de la dérive d'un robot [28].

Les questions relatives à la complexité mécanique, à la consommation d'énergie, à la nature du robot (holonome, omnidirectionnel), sont étudiées dans le chapitre réservé à la présentation de la partie pratique de ce travail de recherche.

1.4 Société multi agents

L'organisation de plusieurs robots mobiles autonomes au sein d'une société est souhaitable, notamment dans les missions de recherche et de sécurisation. La présence de plusieurs robots, travaillant en étroite coordination, peuvent couvrir une plus grande surface en un minimum de temps. L'idée de remplacer un robot unique, aussi puissant soit-il, par plusieurs petits robots organisés en société est un sujet d'actualité, vue sa rentabilité d'un point de vue économique : les petits robots coûtent moins chère qu'un robot puis-

sant, et surtout sont efficacité du fait de la redondance matérielle. On a la garantie que la mission globale sera réalisée même en cas de défaillance d'un robot ou plus.

On utilise le terme *essaim* pour désigner une société de robots, dont les éléments qui la constituent, coopèrent ensemble pour réaliser une tâche unique. Il est inspiré de la nature, notamment de l'organisation des sociétés d'insectes.

Roodney Brooks du M.I.T recommande pour une mission d'exploration de la planète Mars, dans un rapport technique adressé à la NASA et intitulé "Fast, Cheep and out of Control", l'envoi de robots réactifs type fourmi. En plus de l'aspect économique, la perte par destruction de plusieurs robots au cours du transit et de l'atterrissage auront très peu d'impact sur la mission globale [9].

Les équipes de robots, engagées dans des compétitions internationales de football, mettent en évidence les concepts essentiels qu'il faut maitriser pour mettre sur pied une société de robots. Les maitres mots sont : coopération et concurrence, Au sein de la même équipe, chaque robot est en compétition avec les autres robots pour s'accaparer les espaces laissés libres pour les déplacements d'une part, et est obligé de coopérer avec ses partenaires pour défendre son goal, récupérer le ballon et marquer un but d'autre part [23].

Les problèmes liés à l'organisation des robots au sein d'une société dépendent de la nature de la mission à réaliser. Dans notre travail de recherche, on s'est focalisé sur la décontamination de bâtiments publics tels que les musées, les centres d'affaires ; Et la décontamination consiste à sécuriser l'espace de travail en neutralisant d'éventuels intrus. Ce problème, présenté dans la littérature scientifique sous des appellations tels que "le problème de poursuite-évasion" ou "le problème du voleur et du policier", consiste à déterminer le nombre d'agents robots optimal pour sécuriser et décontaminer un espace en un minimum de temps. Il est évident que s'il y a trop de robots par rapport au besoin de la mission, les risques liés à l'interférence dans les missions de chaque agent robot de la société apparaissent et accentuent la pression sur la gestion des ressources communes. Cette redondance matérielle, présence d'un nombre de robots supérieur aux besoins de la mission, rassure sur la continuité de la mission en cas de défaillance d'un robot, mais inquiète aussi en cas de conflits générés par la concurrence liée à l'accès aux ressources communes et peut même mettre en danger l'intégrité physique des agents ainsi que la réalisation de l'objectif final.

La communication est plus que nécessaire, elle est vitale. Chaque agent robot doit savoir en permanence l'état des autres robots de sa communauté et le degré d'achèvement de la mission global. La nature de communications centralisées ou décentralisées et la reconfiguration des missions et des rôles en cas de défaillance d'un élément de la communauté sont des problèmes d'une grande actualité.

Dans [45], l'idée d'octroyer suffisamment d'intelligence à chaque agent de la communauté, augmentant ainsi son degré d'autonomie, afin de lui permettre d'extrapoler l'état des autres agents, est développée, réduisant ainsi les communications entre les agents aux

strict minimum. Ce haut degré d'autonomie, même s'il est accompagné par le risque de voir l'objectif d'un agent être en conflit avec l'objectif du reste de la communauté, permet tout de même aux agents de la communauté d'être en mesure de s'adapter à un monde ouvert.

1.4.1 Société homogène

Les sociétés d'agents homogènes, appelée essaim en référence au modèle biologique d'organisation de la société de fourmis, sont constituées de robots identiques. Ceci a pour avantage de simplifier le coût de fabrication et de faciliter la programmation. Cette organisation sous d'essaim, où l'approche réactive est privilégiée, ont été largement étudiés dans les années 1980.

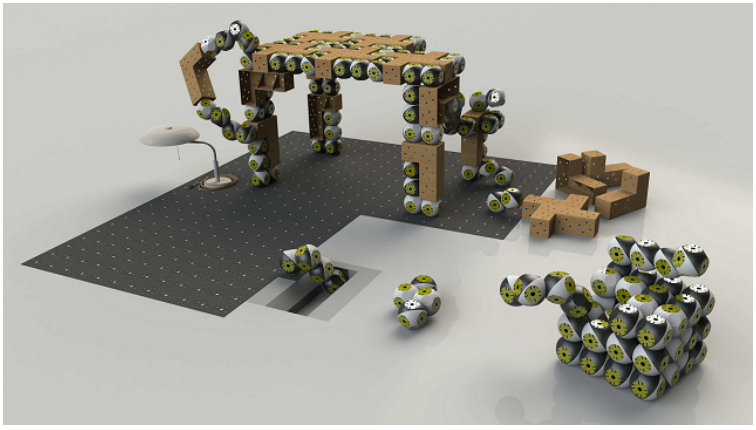


FIGURE 1.3 – Roombots modules se déplaçant à l'intérieur et à l'extérieur de la grille pour former des meubles adaptatifs [52].

Une conférence internationale, intitulée *de l'Animal à l'Animat*, où sont présentés les travaux les plus récents sur la simulation du comportement adaptatif, est organisée annuellement [26].

Dans [50], un algorithme, utilisant un nouvel optimiseur basé sur les techniques "essaim de particules", est conçu pour former des équipes multi agents avec des connaissances qualifiées de niveau zéro. L'algorithme *CCPSO(t)* est appliqué pour former une équipe d'agents pouvant jouer au football de manière simple.

Dans [1], un algorithme distribué pour vérifier si une équipe de robots forme un graphe biconnexe [5]. Ce travail présente aussi des algorithmes pour intégrer ou retirer un robot d'une équipe multi-robots tout en conservant la propriété biconnexe distribué.

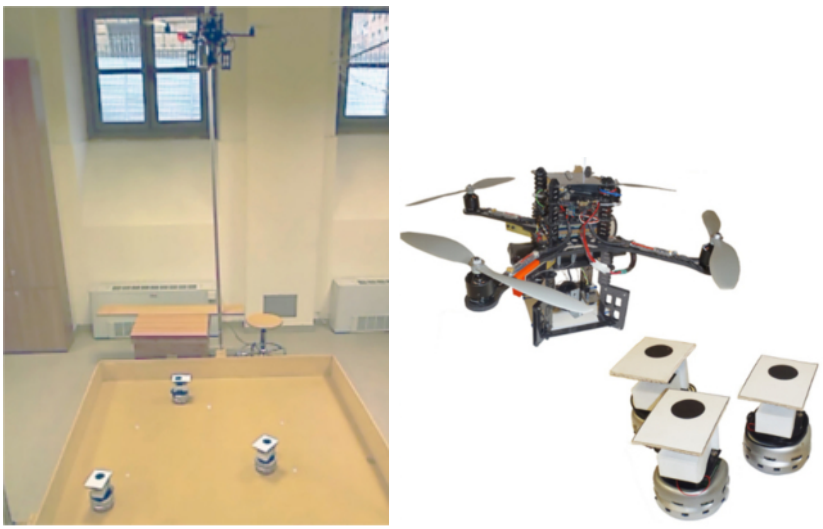
Dans [3], une approche de la conception de contrôleurs pour les robots d'auto-assemblage dans laquelle l'auto-assemblage est initié et régulé par des indices de perception engendrés

5. Un graphe G est biconnexe si, après la suppression de n'importe quel sommet, il est possible de trouver un chemin de n'importe quel sommet vers n'importe quel autre sommet.

par les robots physiques et leurs interactions dynamiques, est illustrée. Plus précisément, un système de commande homogène est présenté, il peut réaliser l'assemblage entre deux modules (deux robots totalement autonomes) en un système auto-reconfigurable mobile sans une hétérogénéité comportementale et morphologique introduites priori.

1.4.2 Société hétérogène

La société d'agents hétérogènes [44, 51, 61] constitue la nouvelle tendance de la robotique mobile autonome. La structure retenue consiste à avoir au moins un robot différent des autres robots de la communauté. Ce robot, plus fort que les autres, joue le rôle de chef d'équipe. Il est doté de suffisamment de capacités et d'intelligence pour diriger les autres robots et cordonner leurs mouvements. Il est évident que la réussite de la mission finale dépend de la capacité de ce super robot à rester disponible durant l'exécution de la mission, car aucun des autres robots de la communauté n'est en mesure de le remplacer en cas de défaillance.



(a) Implémentation des robots hétérogènes.

(b) Robots hétérogènes.

FIGURE 1.4 – Localisation relative et identification dans un système hétérogène Multi-Robot [53].

Dans [53], une méthode de localisation pour un système hétérogène multi-UGV/mono-UAVest développée. Elle exploite le rôle naturel de surveillance des drones. La figure 1.4 représente un système hétérogène constitué d'un robot UAV (type *Pelican*) et de plusieurs petits robots mobiles (type *Khepera III*).

Dans [44], une méthode de conception d'essaims robotiques hétérogènes est présentée pour trouver des solutions à travers un algorithme génétique. Ces robots autonomes peuvent être déployées à des fins telles que la prévention des catastrophes, l'enquête géographique, et de l'exploration sous-marine.

Dans [51], un système qui intègre la navigation autonome, une tâche de direction, la planification des tâches, et une interface utilisateur graphique intuitive pour contrôler plusieurs robots hétérogènes est décrit. Les tests ont montré l'efficacité et la robustesse du système, ainsi que des stratégies de coordination, en particulier.

Dans [61], une nouvelle approche pour cibler l'affectation des équipes hétérogènes de robots, est présentée. Elle va au-delà des algorithmes d'affectation cible existants car elle prend explicitement en compte les actions symboliques. Ces actions comprennent le déploiement et la récupération des autres robots ou des tâches de manipulation. Cette méthode intègre une approche de planification temporelle avec un planificateur traditionnelle basée sur l'optimisation des coûts.

1.4.3 Contrôle et gestion de la société

La gestion de la société d'agents peut être centralisée, où un ordinateur joue le rôle d'un téléopérateur, il collecte les informations à partir des agents et leurs transmet les nouvelles consignes. Dans un système décentralisé, chaque agent robot, en fonction de l'état d'achèvement de la mission finale et de l'état des autres robots, prend ses propres décisions en planifiant sa nouvelle mission ainsi que sa nouvelle trajectoire.

Entre ces deux systèmes, où le premier est complètement centralisé alors que le deuxième est intégralement distribué, se trouvent des systèmes intermédiaires dans lesquels les agents robots reçoivent régulièrement les nouveaux objectifs à réaliser et agissent de manière autonome pour les réaliser.

1.5 Théorie des graphes

Du fait que les environnements de travail dans lesquels vont évoluer les robots organisés en société sont des bâtiments publics tels que les musées, les centres d'affaires et les banques, composés essentiellement de larges couloirs et de vastes salles ou bureaux, on peut utiliser les graphes pour les modéliser en assimilant les coins formés par l'intersection des murs à des sommets et en assimilant la visibilité directe entre les sommets à des arêtes. De la théorie des graphes, on ne présente que les définitions des concepts utilisés pour modéliser les environnements retenus dans ce travail de recherche.

1.5.1 Graphe

Un graphe est un couple $G = (V, E)$ défini par l'ensemble fini $V = \{v_1, \cdots, v_n\}$ dont Les éléments sont appelés sommets ou nœuds (Vertices en anglais), et par l'ensemble fini $E = \{e_1, e_2, \cdots, e_m\}$ dont les éléments sont appelés arêtes ou lignes (Edges en anglais) reliant des paires de sommets de l'ensemble V.

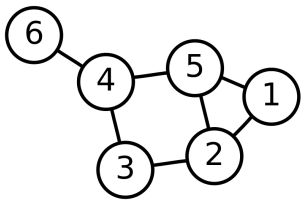


FIGURE 1.5 – Exemple de graphe non orienté

1.5.2 Le degré d'un vertex

Soit le graphe $G = (V, E)$ non vide, l'ensemble des sommets de V voisins d'un sommet v_i, noté par $d(v_i)$ représente son degré (ou sa valence). Il correspond au nombre de paire de sommets de l'ensemble E dans lesquelles le sommet v_i est présent. Il permet de mesurer le nombre de sommets voisins au sommet v_i. Un sommet de degré 0 est un nœud isolé. le nombre $\delta(G) := \max\{d(v) \,|v \in V\}$ mesure le degré maximal du graphe G. Si tous les sommets de G ont le même degré k, alors G est dit k-régulier, ou simplement régulier. A titre d'exemple un graphe 3-régulier est appelé cube.

1.5.3 Chemins et cycles

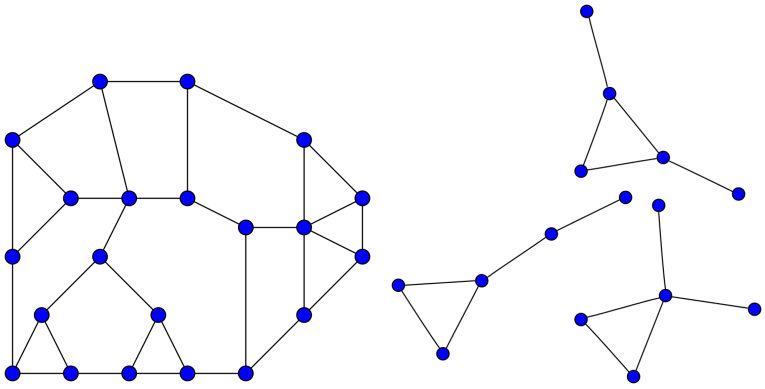
Un chemin est un graphe non vide constitué par une suite ayant pour éléments alternativement des sommets et des arêtes, commençant et se terminant par des sommets.

Le nombre d'arêtes d'un chemin constitue sa longueur, et le chemin P de longueur k est noté P_k.

On se réfère souvent à un chemin en le représentant par la succession naturelle de ses sommets. Un cycle est un chemin dont les sommets extrémités coïncident.

1.5.4 Graphe connexe

Un graphe non vide G est appelé connexe si pour chaque paire de sommets de V il existe un chemin les reliant. Un graphe non connexe se décompose en composantes connexes.



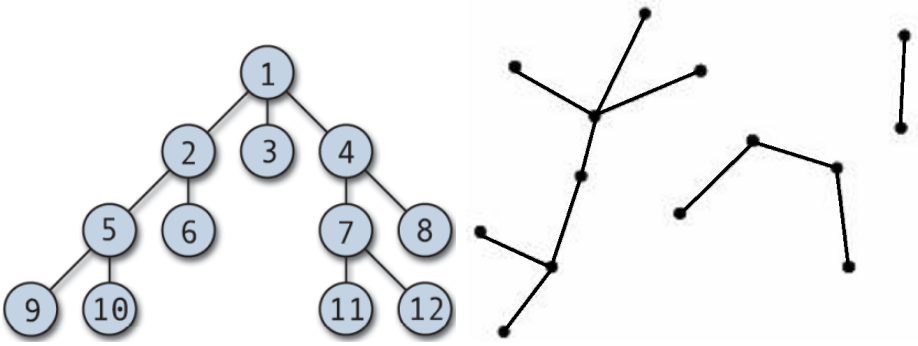
(a) Graphe connexe.

(b) Graphe non connexe, avec trois composantes connexes.

FIGURE 1.6 – Graphes connexes

1.5.5 Arbre et foret

On appelle arbre tout graphe connexe sans cycle et un graphe sans cycle et non connexe est appelée une forêt. Ainsi, une forêt est un graphe dont les composantes sont des arbres. Les sommets de degré 1 dans un arbre sont ses feuilles. Chaque arbre non trivial a une feuille – Considérons, par exemple, les extrémités d'un long chemin. Lorsqu'on enlève une feuille d'un arbre, ce qui reste est encore un arbre.



(a) Exemple d'un arbre.

(b) Exemple de forêt.

FIGURE 1.7 – Exemple d'un arbre et de forêt

Figure 1.7(a) présente un arbre ayant (1) comme racine, (2), (4), (5), (7) comme nœuds internes et (3), (6), (8), (9), (10), (11), (12) comme feuilles.

Cette définition est adaptée à partir de [16]

Définition 1 *Les assertions suivantes sont équivalentes pour un graphe T :*

— *T est un arbre ;*

— *deux sommets de T sont reliés par un chemin unique en T ;*

— *T est peu connecté, à savoir T est connecté, mais $T-e$ est déconnecté pour chaque arête ;*
— *T est au maximum acyclique, à savoir T ne contient pas de cycle, mais $T+xy$ l'est, pour deux sommets x, y non adjacentes appartenant à T*

1.5.6 Cycle

Dans un graphe non orienté, un cycle est une suite d'arêtes consécutives (chaine) dont les deux sommets extrémités sont identiques. Le terme de cycle désigne parfois aussi le graphe cycle C_n constitué d'un cycle élémentaire de longueur n.

Si la chaine est élémentaire, c'est-à-dire ne passe pas deux fois par un même sommet, alors on parle de cycle élémentaire. Un cycle élémentaire ne contient pas d'autre cycle. Dans un cycle élémentaire, le degré de tous les sommets est égal à deux.

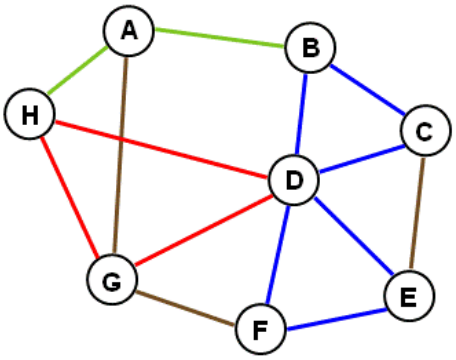


FIGURE 1.8 – Exemple de cycles, le cycle rouge est élémentaire, le cycle bleu ne l'est pas. La chaine verte n'est pas fermée et ne forme donc pas un cycle.

1.5.7 Clique

Une clique d'un graphe non orienté est, en théorie des graphes, un sous-ensemble des sommets de ce graphe dont le sous-graphe induit est complet, c'est-à-dire que deux sommets quelconques de la clique sont toujours adjacents.

Une clique maximum d'un graphe est une clique dont le cardinal est le plus grand (c'est-à-dire qu'elle possède le plus grand nombre de sommets).

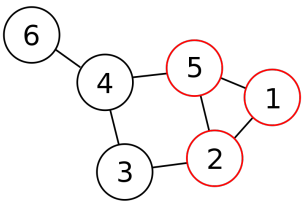
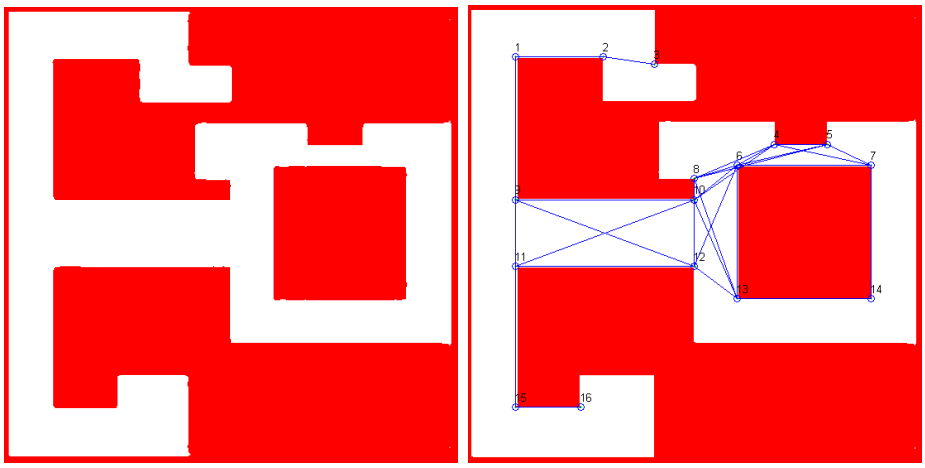


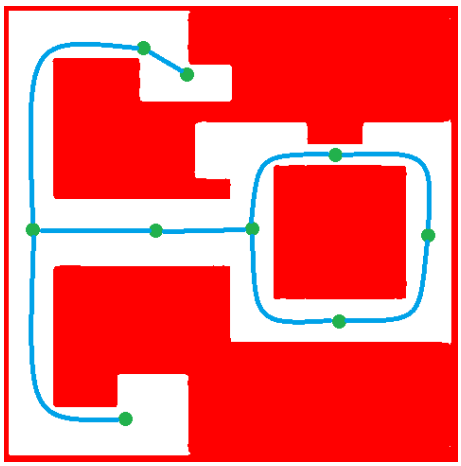
FIGURE 1.9 – Exemple de graphe possédant une 3-clique (en rouge).

1.5.8 Utilisation de la théorie des graphes

L'environnement où évoluent les robots est décomposé en régions libres d'obstacles et d'autres parsemées d'obstacles. La figure 1.10(a)) est un exemple d'environnement où les espaces libres pour les déplacements (couloirs) sont représentés en noir et les espaces constitués d'obstacles et interdits à la navigation (murs) sont représentés en rouge. La figure 1.10(b) montre le graphe obtenu en définissant un sommet comme étant l'intersection des murs d'un couloir faisant un angle supérieurs à π (définition 2) et la visibilité directe entre deux sommets comme une arête (définition 3), il est appelé graphe de visibilité. Il peut être simplifié en remplaçant les sous-graphes complets (cliques) par des sommets et obtenir un graphe de visibilité simplifié tel que représenté dans la figure 1.10(c).



(a) Exemple d'environnement (b) Graphe de Visibilité équivalent.



(c) Graphe de Zones équivalent.

FIGURE 1.10 – Exemple d'utilisation de la théorie de graphes

Définition 2 *Un sommet de V est l'intersection de deux murs formant un angle supérieur à π.*

Définition 3 *Une arête représente la visibilité directe entre deux sommets.*

1.6 Poursuite-évasion

Le problème de Poursuite-évasion, appelée aussi problème du policier et du voleur, est un sujet de recherche qui continue de susciter de l'intérêt, il est toujours d'actualité [5, 27, 29, 30, 32, 38, 37, 54, 58, 59, 60]. Il permet de mettre en évidence la puissance de représentation des graphes, dans les problèmes de recherche d'intrus, en décomposant l'espace de recherche en zones décontaminées et en zones contaminées.

Pour les lieux publics sensibles tels que les galeries d'art, la théorie des graphes permet de déterminer de manière optimale le nombre d'agents surveillants nécessaires à la sécurisation des lieux [13]. Dans une configuration dynamique, telles que les opérations de recherche de blessés ou de sauvetage dans des environnements qui sont devenus connus partiellement à la suite d'une catastrophe naturelle, le nombre d'agents secouristes n'est pas connu d'avance, il est optimisé au fur et à mesure de la progression de la mission. Une riche littérature est consacrée à ce thème de recherche [27, 38, 37, 58, 21, 17]. Pour des environnements inconnus, les travaux de recherche développés dans [30, 59, 60, 17] mettant l'accent sur l'utilisation de capteurs permettant d'appréhender au mieux la découverte progressive de l'environnement et de gérer dynamiquement le nombre d'agents en fonction de l'évolution de la mission et de la progression de la découverte de l'environnement. La communication entre les agents est un élément clé de la réussite de la mission [32].

Dans [35], un exemple, dans lequel un poursuivant "aveugle" est à la recherche d'un fraudeur se déplaçant sur un réseau routier avec une vitesse connue et se dirigeant vers un ensemble de sommets, est considéré. Un ensemble de capteurs au sol type (UGS) ont été placés sur les réseaux routiers permettant de détecter le passage du fraudeur à leur niveau. Un contrôleur optimal a été développé. Dans la Figure 1.11, un réseau routier illustratif est représenté. Les routes sont indiquées en rouge (les flèches indiquent la direction de déplacement) et les UGS numérotées sont des cercles bleus.

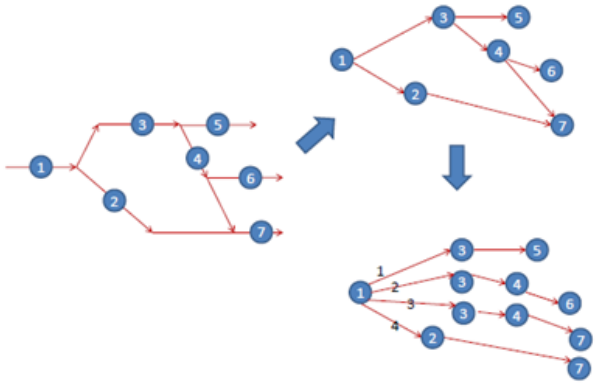


FIGURE 1.11 – Réseau de routes, graphe UGSs, et 4 chemins possibles [35]

Dans [7], deux résultats généraux sur les jeux "poursuite-évasion" dans des espaces topologiques sont démontrés. Tout d'abord, il est montré qu'un poursuivant a une stratégie

gagnante dans tous les espaces d'Hadamard sous le critère de capture restrictive.

[40] étudie le problème de poursuite-évasion dans sa version "policiers et voleurs" sur les graphes de visibilité en forme de polygones. Chaque joueur dispose d'une information complète sur l'emplacement de l'autre joueur, les joueurs se relaient, et le voleur est capturé quand un agent policier arrive au même point que le voleur.

Le chapitre 2 est dédié à la présentation du problème de poursuite évasion dans des environnements connus et des environnements partiellement connus.

1.7 Modélisation des systèmes à événements discrets

Rappelons qu'un système à événements discrets (ou *SED*) est caractérisé par son comportement dynamique qui se traduit par le changement d'état vers un autre état suite à l'occurrence d'un événement. Les événements peuvent être de deux types : l'événement de temps est un événement uniquement déterminé par rapport à la variable temps ; l'événement d'état dépend des conditions d'évolution du système et s'obtient par des règles.

Pour pouvoir exploiter au mieux les potentialités d'un système à événements discrets, il faut choisir un modèle.

Un modèle est une approximation, une vue partielle plus ou moins abstraite de la réalité afin de l'appréhender plus simplement. Il est établi pour un objectif donné. Un modèle est donc subjectif puisqu'il est établi en fonction des objectifs, du jugement, de la nature et de la qualité des informations dont dispose le concepteur. Il peut être exprimé par des mathématiques, des symboles, des mots, etc. [10].

Une description et une comparaison des différents outils de modélisation utilisés pour l'évaluation des performances des systèmes à événements discrets est présentée dans les travaux de H. Camus [11].

Les deux classes de modèles habituellement utilisées pour représenter les systèmes à événements discrets sont les réseaux de Petri et les machines à états finis.

1.8 Les réseaux de Petri

Les Réseaux de Petri avec extensions portant sur les arcs inhibiteurs sont utilisés dans ce travail pour d'une part modéliser le processus de décontamination d'un environnement, préalablement décomposé en zones à l'aide de la théorie des graphes, et d'autre part valider le nombre optimal d'agents robots nécessaires à la réussite de la mission. De même les Réseaux de Petri avec extensions portant sur les exigences temporelles sont utilisés pour modéliser les échanges entre les agents robots au cours de leur mission et détecter la défaillance temporaire ou permanente d'un agent. On présente de manière succincte les Réseaux de Petri classiques, puis les réseaux de Petri avec extensions portant sur les

arcs inhibiteurs, et enfin les Réseaux de Petri avec extensions portant sur les exigences temporelles.

1.8.1 Les Réseaux de Petri classiques

Un Réseau de Petri (RdP) est un graphe orienté comprenant deux types de sommets :
— Les **places** ◯
— Les **transitions** ▬
Ils sont reliés par des arcs orientés ↗ . Un arc relie soit une place à une transition, soit une transition à une place jamais une place à une place ou une transition à une transition. Tout arc doit avoir à son extrémité un sommet (place ou transition). Un exemple de Réseaux de Petri est représenté (figure 1.13). A chaque place et transition, un nom peut être associé : par exemple, les places sont nommées $P1$, $P2$, $P3$ et $P4$ et les transitions $T1$, $T2$ et $T3$. $T1$ est reliée à $P1$ par un arc orienté de $T1$ vers $P1$: on dit que $P1$ est en sortie ou en aval de $T1$. $P1$ est reliée à $T2$ par un arc orienté de $P1$ vers $T2$: on dit que $T2$ est en sortie de $P1$. De même, on peut dire que $P1$ est en entrée ou en amont de $T2$. La place $P3$ est en entrée de $T1$.

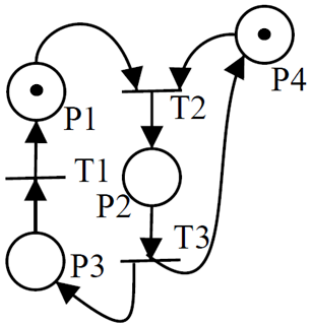


FIGURE 1.12 – Exemple d'un réseau de Petri marqué

Une place correspond à une variable d'état du système qui va être modélisé et une transition à un événement et/ou une action qui va entraîner l'évolution des variables d'état du système. A un instant donné, une place contient un certain nombre de marques ou jetons qui va évoluer en fonction du temps : il indique la valeur de la variable d'état à cet instant. Quand un arc relie une place à une transition, cela indique que la valeur de la variable d'état associée à la place influence l'occurrence de l'événement associé à la transition. Quand un arc relie une transition à une place, cela veut dire que l'occurrence de l'événement associé à la transition influence la valeur de la variable d'état associée à la place.

Définition 4 *(unmarked Petri net [62]) Un réseau de Petri non marqué est un 4-uplet* (P, T, F, V) *de telle sorte que*

1. P *et* T *sont des ensembles finis avec* $P \cap T = \phi$ *et* $P \cup T = \phi$.

2. *F est une relation d'arité 2 avec $F \subseteq (P \times T) \cup (T \times P)$.*

3. *$V : F \to N^+$.*

Définition 5 *(Petri net [62]) Un réseau de Petri marqué est un 5-uplet $N = (P, T, F, V, M_0)$ (court : Petri net) tels que*

1. *(P, T, F, V) est un réseau de Petri non marqué.*

2. *$M_0 : P \to N$ est le marquage initial.*

Définition 6 *(marquage [62]) Soit P l'ensemble des places d'un réseau de Petri N. Un marquage dans N est une fonction totale $m : P \to N$.*

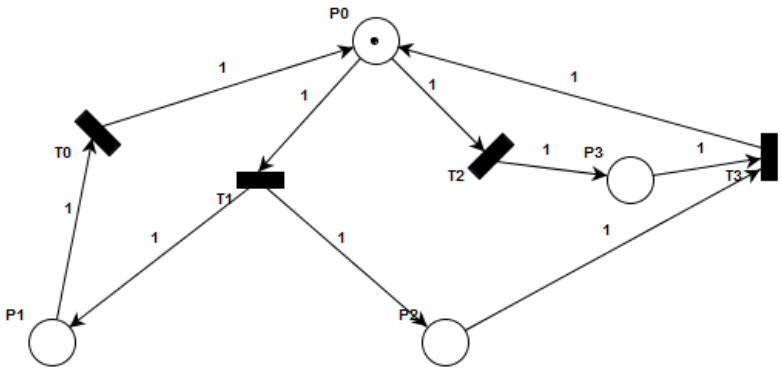


FIGURE 1.13 – N_1 est un réseau de Petri avec marquage initial $M_0 = (1, 0, 0, 0)$

On appelle marquage M d'un Réseau de Petri le vecteur du nombre de marques dans chaque place : la i^e composante correspond au nombre de marques dans la i^e place. Il indique à un instant donné l'état du RdP. Par exemple, le marquage du RdP présenté figure 1.13 est donné par :

$$M = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

On appelle marquage initial, noté M_0, le marquage à l'instant initial ($t = 0$).

Une pré-place d'une transition est une place qui est directement liée à la transition considérée par un arc dirigé de la place vers la transition. Si l'arc est à la direction opposée, cette place est une post-place de la transition. L'ensemble des pré-places de la transition t notée $\bullet t := \{p | p \in P \wedge (p, t) \in F\}$. L'ensemble $t\bullet := \{p | p \in P \wedge (t, p) \in F\}$ est l'ensemble des post-places de t. De manière analogue, pour chaque place p l'ensemble $\bullet p := \{t | t \in T \wedge (t, p) \in F\}$ désigne l'ensemble des pré-transitions de p et l'ensemble $p\bullet := \{t | t \in T \wedge (p, t) \in F\}$ désigne l'ensemble des post-transitions p.

Si tous les multiplicités des arcs d'un réseau de Petri sont 1, la règle de tire étendue de façon consistante. Les conditions d'un événement sont remplies si, pour chaque place, le nombre de jetons qu'il détient n'est pas plus petit que la multiplicité de l'arc de cette place à la transition respective. Après un événement a eu lieu, tous les post-places de la transition obtient le nombre de jetons équivalent à la multiplicité de l'arc de la transition vers l'après-place respective. Une transition t peut donc se déclencher si le réseau de Petri est dans un marquage m, qui attribue au moins autant de jetons que t a besoins de chaque pré-place de t. Toutes les conditions préalables peuvent être considérées comme remplies. Ce nombre minimum de jetons nécessaires sur les places est définie par le W^- :

$$W^- (p) := \begin{cases} V (p,t) , si (p,t) \in F \\ 0, si (p,t) \notin F \end{cases} \tag{1.1}$$

De façon analogue le marquage W^+ décrit le nombre de jetons qui sont ajoutés à chaque place lors de le tir de t :

$$W^+ (p) := \begin{cases} V (t,p) , si (t,p) \in F \\ 0, si (t,p) \notin F \end{cases} \tag{1.2}$$

La différence de jetons sur les places après le tir des transitions T représente le marquage $\Delta W := W^+ - W^-$

En outre, nous considérons le réseau de Petri $N = (P, T, F, V, M_0)$ avec $T = \{t_1, \cdots, t_n\}$ et $P = \{p_1, \cdots, p_m\}$. La matrice $C_N = (c_{ij})$ à m lignes et n colonnes et

$$c_{ij} := \Delta W_j (p_i)$$

Que l'on appelle la *matrice d'incidence* de N.

Exemple : Considerons l'exemple dans la figure 1.13, Sa matrice d'incidence C_{N_1} est :

$$C_{N_1} = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \end{array} \begin{array}{c} \begin{array}{cccc} \Delta t_1 & \Delta t_2 & \Delta t_3 & \Delta t_4 \end{array} \\ \begin{pmatrix} 1 & -1 & -1 & 1 \\ -1 & 1 & 0 & 0 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & -1 \end{pmatrix} \end{array} \tag{1.3}$$

Nous pouvons maintenant présenter les notions de tir :

Définition 7 *(enabled [62]) Soit $N = (P, T, F, V, m_0)$ un réseau de Petri et soit m un marquage dans N. une transition $t \in T$ est active dans m si s'applique que : $t^- \leq m$.*

Définition 8 *(firing [62]) Soit $N = (P, T, F, V, m_0)$ un réseau de Petri et soi m un marquage dans N. une transition $t \in T$ peut tirer dans m (notation : $m \xrightarrow{t}$), si t est*

activée dans m. Après le tire de t le réseau de Petri est dans le marquage m′ (notation : $m \xrightarrow{t} m'$*) avec* $m' := m + \Delta W$

Cette règle de tir définit le tir d'une seule transition, c'est-à-dire, c'est une règle de tir mono. Il est également possible de définir une règle de tir d'ensemble (étape) de transitions. Cette règle de tir, appelé aussi *séquence de tir*, est habituellement utilisée dans les réseaux de Petri en fonction du temps.

On note le changement d'un réseau de Petri à partir du marquage m en m' par tir de la transition t par $m \xrightarrow{t} m'$. La relation $\rightarrow$ qui est défini par la règle de tir est appelée la relation de tir.

Une nouvelle règle de base dans la théorie des réseaux de Petri est la notion d'un marquage accessible. Pour l'introduire, nous définissons d'abord la séquences de tir :

Définition 9 *(firing sequence [62]) Soit* $N = (P, T, F, V, M_0)$ *un réseau de Petri, soit m un marquage en N et soit* $\sigma = t_1 \cdots t_n$ *une suite de transitions.* σ *tire de m à m′ en N (courte :* $m \xrightarrow{\sigma} m'$*) s'il s'avère que :*

- ***Base*** $\sigma = \epsilon \rightarrow m' := m$
- ***Pas*** $\sigma = t_1 \cdots t_n t_{n+1}$

Il y a un marquage m″ dans N, avec

$$m \xrightarrow{t_1 \cdots t_n} m'' \text{ et } m'' \xrightarrow{t_{n+1}} m'$$

σ est appelée une séquence de tir de m en N s'il y a un marquage m′ tel que σ tire de m à m′. Une séquence de tir σ de m_0 *en N est généralement appelée simplement une séquence de tir.*

1.8.2 Réseaux de Petri à arcs inhibiteurs

La logique sous-jacente au déclenchement d'une transition est basée sur la présence d'un ou de plusieurs jetons dans les places. S'il n'y a pas de jeton, il n'y a pas de déclenchement. Il peut être intéressant de disposer d'un mode de déclenchement inverse, selon lequel la transition serait franchissable si la place ne contient aucun jeton. On pourrait ainsi, par exemple, exprimer le traitement de commandes de produit, avec une transition pour le cas où il y a assez de produit en stock et une autre pour le cas où il n'y a plus de produit.

Les arcs inhibiteurs sont conçus pour corriger cela. Ils permettent le déclenchement de la transition uniquement si la place qu'ils relient à la transition est vide. Ils sont représentés graphiquement comme suit :

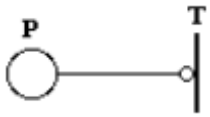


FIGURE 1.14 – Représentation graphique d'un arc inhibiteur

La règle de déclenchement d'une transition devient : une transition T est déclenchable si :

— Toutes les places appartenant à son entrée négative $H(T)$ sont vides ;
— Toutes les places appartenant à son entrée positive $W^+(T)$ contiennent au moins autant de jetons que le poids des arcs par lesquels elles sont reliées à la transition.

Exemple : Ainsi, dans l'exemple ci-après, pour déclencher $T1$, il faut un jeton dans $P1$ et trois dans $P2$; il faut aussi que $P3$ soit vide.

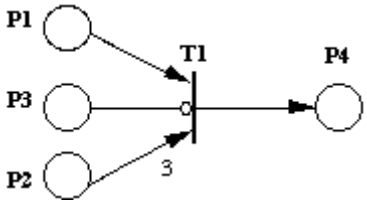


FIGURE 1.15 – Exemple de RdP avec arc inhibiteur

1.8.3 Les Réseaux de Petri avec exigences temporelles

Réseaux de Petri temporisés (RdPTs)

Aucune durée n'est liée au franchissement des transitions et/ou au temps de séjour des marques dans les places en ce qui concerne les RdPAs. Cependant il y a beaucoup de systèmes à événements discrets dont l'évolution dépend du temps. D'autre part la notion de temps est capitale lorsque l'on veut évaluer les performances ou étudier les problèmes d'ordonnancement d'un système dynamique [43]. La nécessité de modéliser et d'étudier de tels systèmes a donné naissance aux RdPs temporisés (RdPTs) [33].

Définition 10 *Un* p-RdPT *est un doublet* $< R, Tempo >$ *tel que :*

— *R : un RdP autonome marqué*
— *Tempo : une application de P vers* Q^+ *(ensemble des rationnels positifs ou nuls).*

$Tempo(p_i) = d_i$: temporisation associée à la place p_i.

La sémantique de l'application Tempo est que les marques doivent rester dans la place p_i au moins le temps d_i associé à cette place. d_i représente donc la durée d'indisponibilité de la marque pour la validation des transitions.

Définition 11 *Un état est un doublet* $< M, I >$ *où*

— *M est une application de marquage, assignant à chaque place du réseau un certain nombre de marques* ($\forall p \in P, M(p) \geq 0$) ;
— *I est une application de temps d'indisponibilité, assignant à chaque marque k dans la place* p_i *un temps* θ_i^k, θ_i^k *est la durée qui reste à la marque k pour terminer son temps de séjour minimal dans la place* p_i.

Notons que la transition peut être validée au sens des RdP autonomes et qu'elle peut ne pas être franchissable à cause des temporisations. Une transition est franchissable si elle est validée au sens des RdP autonomes et si les marques qui la valident sont disponibles.

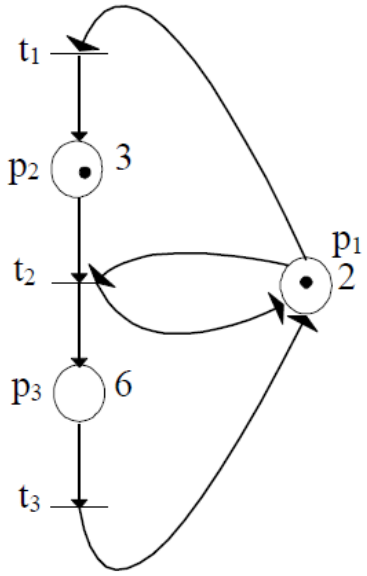


FIGURE 1.16 – Exemple de réseau de Petri Temporisé

Les réseaux de Petri t-temporels (t-RdPs)

Les *RdPTs* ne permettent que de spécifier une durée minimale de traitement. Bien qu'ils permettent de modéliser un large nombre de systèmes manufacturiers [49], ils ne sont pas capables de modéliser des systèmes pour lesquels la durée de traitement est incluse entre une valeur minimale et une valeur maximale [33].

Les RdP t-temporels (t-RdPs) sont destinés principalement à l'étude des systèmes de télécommunication dont les évolutions dépendent des contraintes de type temps de réponse (time-out) [41, 6].

Définition 12 *Un t-RdP est un doublet* $< R; IS >$ *où*
— *R est un réseau de Petri marqué*
— $IS : T \rightarrow (Q^+ \cup 0) \times (Q^+ \cup \infty)$ *(Q^+ est l'ensemble des nombres rationnels positifs)*
— $t_i \rightarrow IS_i = [a_i, b_i]$ *avec* $0 \leq a_i \leq b_i$

IS_i définit un intervalle statique (pour le différencier de l'intervalle dynamique qui sera défini après). Une transition t_i doit être sensibilisée (validée) pendant le délai minimum a_i

avant de pouvoir être tirée (franchie) et ne peut rester validée au-delà du délai maximum b_i sans être tirée. Le tir d'une transition est de durée nulle.

Définition 13 *L'état d'un t-RdP est défini par une paire $E = (M, I)$, contenant :*

- *M, une application de marquage, assignant à chaque place du réseau un certain nombre de marques ($\forall p \in P, M(p) \geq 0$) ;*
- *I, une application intervalle de tir, associant à chaque transition du réseau l'intervalle de temps dans lequel elle peut être tirée ($t \in T, I(t) = \Phi$ si et seulement si t est non-validée). Les intervalles de tir dans l'état courant peuvent différer de ceux assignés initialement (les intervalles statiques) aux transitions du réseau. Pour cela ils seront appelés "intervalles dynamiques".*

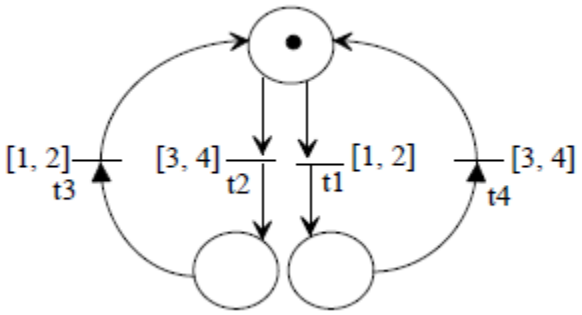


FIGURE 1.17 – Exemple de réseau de Petri t-temporel

[63, 64] présente un cadre de représentation original basé sur les réseaux de Petri pour décrire les comportements du robot et multi-robots. Le formalisme de Plan à réseau de Petri (Petri Net Plan ou PNP) permet la description du niveau élevé d'interactions complexes d'action qui sont nécessaires dans la programmation des robots cognitifs : actions non-instantanées, de détection et d'actions conditionnelles, les échecs d'action, des actions simultanées, les interruptions, la synchronisation d'action dans un contexte multi-agent. Il montre ainsi comment ce cadre est capable de décrire des plans efficaces pour les agents robotiques qui inhibent un environnement dynamique, partiellement observable et imprévisible.

Dans [12] un réseau de Petri haut niveau de système multi Agent (appelé HMAP) a été proposée qui est capable de décrire, d'analyser et de modéliser la dynamique de tels systèmes multi agents qui sont caractérisés comme des systèmes à base d'agents asynchrones, distribués, parallèles et non déterministes.

1.9 Machines à états finis

Un automate fini[6] (on dit parfois machine à états finis), est une machine abstraite utilisée en théorie de la calculabilité et dans l'étude des langages formels. Un automate

6. en anglais "finite state automaton" ou "finite state machine" (FSA ou FSM)

est constitué d'états et de transitions. Son comportement est dirigé par un mot fourni en entrée : l'automate passe d'état en état, suivant les transitions, à la lecture de chaque lettre de l'entrée. Un automate fini possède un nombre fini d'états distincts : il ne dispose donc que d'une mémoire bornée.

Une machine à états peut être représentée graphiquement par un graphe étiqueté. Les états sont des cercles (ou des ellipses) et les transitions sont des flèches orientées de leurs états de départ vers leurs états de destination.

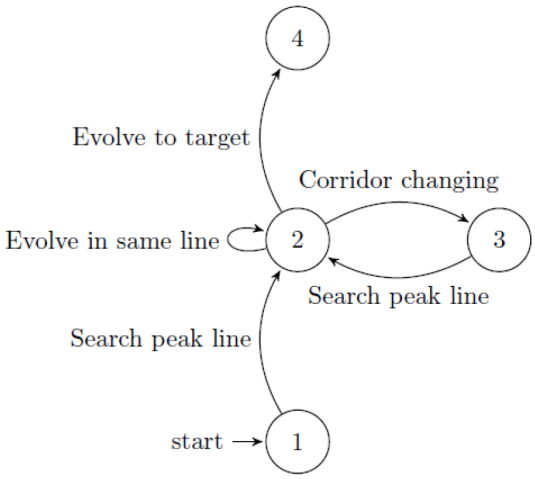


FIGURE 1.18 – Graphe de navigation selon [19].

Définition 14 *Formellement, un Automate Fini Déterministe (DFA) est un quintuplet :* $(S, \sum, T, s, A)$

- *un alphabet* $(\sum)$
- *un ensemble d'états* (S)
- *une fonction de transition* $(T : S \times \sum \rightarrow S)$.
- *un état de départ* $(s \in S)$
- *un ensemble d'états acceptant* $(A \subseteq S)$

Pour construire un modèle à événements discrets à l'aide d'une machine à états, il faut énumérer explicitement les états dans lesquels peut se trouver le système étudié. Ce qui présente une tâche difficile pour le concepteur du modèle. Il faut donc prévoir tous les états possibles à partir d'un scénario prévu et du matériel disponible.

1.10 Conclusion

Ce chapitre donne une idée générale sur les informations de base nécessaires pour comprendre le travail envisagé, tel que la poursuite-Évasion et les agents intelligents.

La prochaine partie expliquera le problème de poursuite évasion et les technique de résolution de ce problème.

Chapitre 2

Poursuite-évasion et théorie des graphes

2.1 Introduction

Dans ce chapitre, on aborde le problème de la décontamination d'un environnement, représentant un espace public constitué de couloirs et de salles. La décontamination consiste à assurer la sécurité des lieux en recherchant et en neutralisant d'éventuels intrus. La société d'agents est engagée dans un problème connu dans la littérature scientifique sous l'appellation de "Problème de Poursuite-Évasion". La théorie des graphes permet d'établir un modèle topologique de l'environnement et les réseaux de Petri permettent la validation du nombre d'agents nécessaires à l'opération de décontamination.

2.2 Problème de Poursuite-Évasion

Dans la théorie de jeu, le problème de poursuite-évasion est considéré comme un jeu se composant de trois parties : des joueurs, des actions et une fonction de coût. Les joueurs sont les agents nettoyeurs d'une part et les intrus d'autre part. Les actions sont les déplacements des antagonistes du jeu dans l'environnement, et la fonction coût intègre le temps mis par les agents à neutraliser les intrus et le nombre d'agents nécessaire pour le réaliser [27, 38, 37].

Les joueurs antagonistes de ce jeu reçoivent des informations différentes du fait que chaque camp est doté de moyens de perception différents ; Dans certains travaux l'avantage octroyé aux intrus, est poussé jusqu'à leurs accorder une meilleure connaissance de l'environnement, une idée précise des positions des agents nettoyeurs et des informations sur leurs stratégies de déplacements.

Les actions que peuvent entreprendre les joueurs dépendent de l'environnement et du modèle de déplacement associé à chacun d'eux. L'état d'une action ne doit pas produire

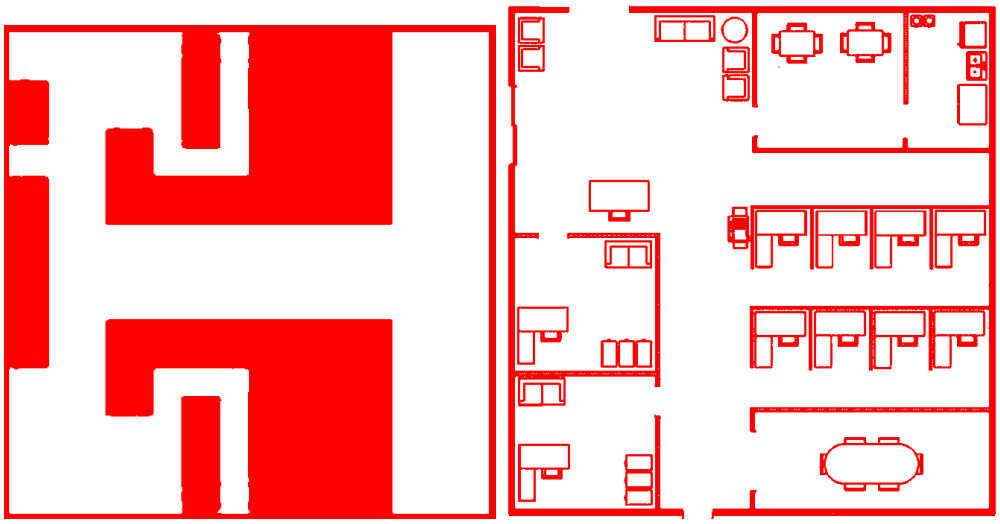
de collisions avec des obstacles présents dans l'environnement.

L'objectif des joueurs dans le jeu de poursuite-évasion est de trouver des politiques qui optimisent leurs fonctions coûts. Si pour les agents robots, l'objectif final est de neutraliser l'intrus en minimisant le temps d'exécution de la politique de recherche et le nombre d'agents engagé dans cette opération, pour les intrus l'objectif est tout autre, il consiste à éviter à se faire découvrir et ce en restant le plus longtemps possible loin de la visibilité directe des agents nettoyeurs.

2.3 Modélisation topologique de l'environnement

2.3.1 Graphe de visibilité

Les environnements retenus dans ce travail de simulation sont des espaces publics composés de couloirs et de salles ou de bureaux. La figure 2.1 représente des exemples d'environnements où les espaces libres pour les déplacements sont représentés en blanc et les espaces non accessibles (obstacles) sont représentés en rouge.



(a) Environnement 1. (b) Environnement 2.

FIGURE 2.1 – Exemples d'environnements

L'étape suivante consiste à modéliser cet environnement par un graphe. Dans notre application les environnements sont enregistrés sous forme d'image et des traitements simples permettent d'en extraire les contours (murs de couleur verte sur la figure 2.2). En utilisant la définition 2 établie dans le chapitre précédent (un sommet est l'intersection de deux murs formant un angle supérieur à π), on établit tous les nœuds du graphe. La figure 2.2 représente les nœuds obtenus à partir du premier environnement de la figure 2.1(a).

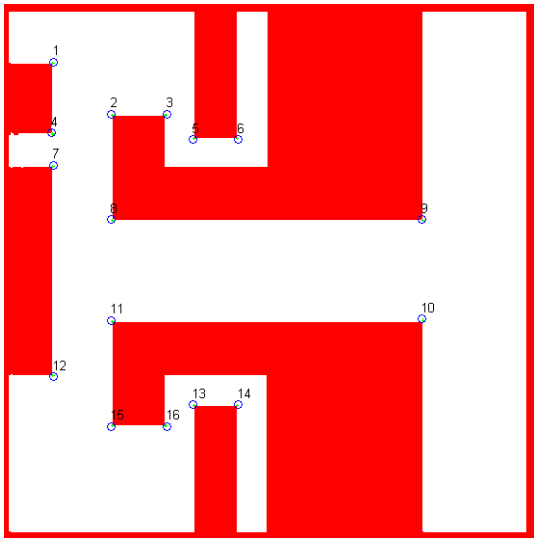


FIGURE 2.2 – Détection de contour et établissement des nœuds du graphe

En établissant l'ensemble des paires de nœuds et en utilisant la définition 3 présentée dans le chapitre précédent (une arête représente la visibilité directe entre deux sommets), le graphe de visibilité généré est représenté dans la figure 2.3. C'est le modèle topologique du premier environnement de la figure 2.1(a).

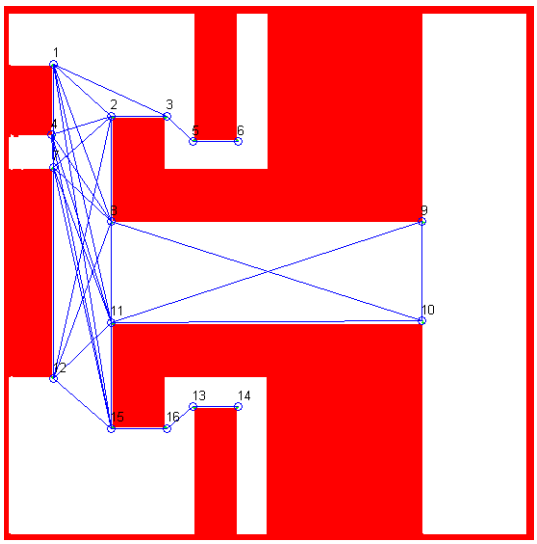


FIGURE 2.3 – Graphe de visibilité

Le graphe de visibilité représenté dans la figure 2.3 est le modèle topologique de l'environnement de la figure 2.1(a). Ce graphe est l'élément le plus important dans cette partie de travail, et toutes les stratégies de recherche sont basées sur des graphes établis à partir de ce dernier.

2.3.2 Graphe de zones

Cette étape consiste à extraire, à partir du graphe de visibilité, le degré de chaque sommet (nombre de sommets adjacents), les sous graphes complets appelés *cliques* ou zones, d'établir le graphe connexe des zones, et de déterminer celles qui sont critiques.

Les fonctions ou algorithmes utilisés pour établir le graphe de zones sont :

1. La fonction **degré-sommet** permet d'établir le tableau 2.1 représentant le degré d'adjacence de chacun des sommets du graphe.

sommet	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
degré	8	8	3	7	2	1	7	9	3	3	9	7	2	1	8	2

TABLE 2.1 – Degré des sommets du graphe de vibilité

2. La fonction **décompose-graphe**, permettant de décomposer le graphe de visibilité en sous graphes complets de même degré et interconnectés, est basée sur l'algorithme développé par Bron et Kerbosch en 1973 et qui demeure jusqu'à aujourd'hui l'un de plus rapides et les plus utilisés des algorithmes de détection de cliques dans un graphe [55].

Les étapes de l'algorithme sont présentées dans les lignes de l'algorithme 2.1 (extrait de [14]) :

Algorithme 2.1 Algorithme de recherche des cliques maximales

BronKerbosch1(R, P, X)
if P et X son vides **then**
  retourner R comme clique maximale
end if
for all sommet v de P **do**
  BronKerbosch1($R \cup \{v\}$, $P \cap N(v)$, $X \cap N(v)$)
  $P := P \setminus \{v\}$
  $X := X \cup \{v\}$
end for

avec les trois ensembles R, P, et X, cet algorithme trouve les cliques maximales contenant tous les sommets dans R, quelques sommets dans P, et aucun sommet dans X.

On obtient huit (08) sous-graphes complets interconnectés dont les degrés sont consignés dans le tableau 2.2 :

Zones ou clique	Zone 1	Zone 2	Zone 3	Zone 4	Zone 5	Zone 6	Zone 7	Zone 8
Liste des sommets	{1,2,4,7, 8,11,12, 15}	{1,2,3}	{8,11,9, 10}	{3,4}	{5,6}	{15,16}	{16,13}	{13,14}
Degré de la clique	3	2	1	2	1	2	2	1

TABLE 2.2 – Les cliques du graphe de visibilité

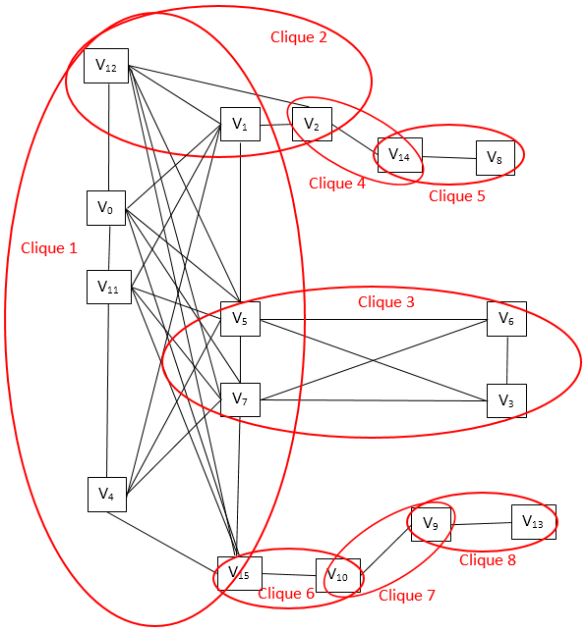


FIGURE 2.4 – Décomposition de d'un environnement de la Figure 2.1(b) en cliques ou zones.

A partir de la figure 2.4, on obtient un nouveau graphe, appelé graphe de zones, dont les sommets sont les cliques et les arêtes représentent la connexion entre les zones. Elle est traduite par les sommets appartenant à deux cliques adjacentes.

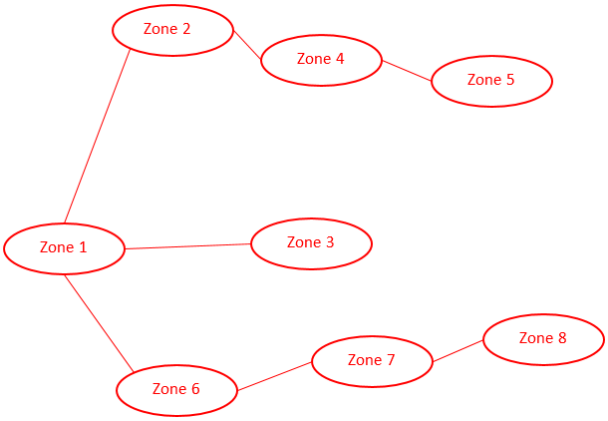


FIGURE 2.5 – Graphe de zones

Par exemple l'arête reliant la zone1 à la zone2 est matérialisée par les sommets communs à la $clique_1$ et à la $clique_2$: $Clique_1 \cap Clique_2 = \{V_1, V_2\}$

2.3.3 Degré d'une zone

On définit le degré d'une zone de manière identique à celui d'un sommet, il correspond au nombre de zones adjacentes auxquelles elle est reliée.

— Une zone de degré 1 est une *feuille*, c'est une zone terminale.
— Une zone de degré 2 est un élément d'un *chemin*, c'est une zone de passage appelée couloir.
— Une zone de degré supérieur à 2 est une *zone critique*, elle est reliée à plus de deux zones ou chemins.

2.4 Techniques de recherche

2.4.1 Introduction

Les algorithmes présentés dans cette partie traitent de la décontamination des environnements connus et des environnements partiellement connus ou inconnus. Dans le cas des environnements connus, deux techniques de recherche d'intrus sont étudiées ; La première méthode est simple qu'on peut qualifier de méthode **naïve**, alors que la deuxième est plus élaborée, elle permet d'optimiser le nombre d'agents robots engagés dans l'opération de recherche en manipulant une fonction de *fitness*. Dans le cas des environnements inconnus, deux méthodes sont présentées : la première se base sur une stratégie de recherche directe appelée technique **forward**, alors que la deuxième technique se base sur une stratégie de va-et-vient appelée technique **forward-backward**.

2.4.2 Recherche dans un environnement connu

La figure 2.6 représente l'environnement étudié avec son graphe de visibilité et sa décomposition en cliques, alors que la figure 2.7 représente son graphe de zones.

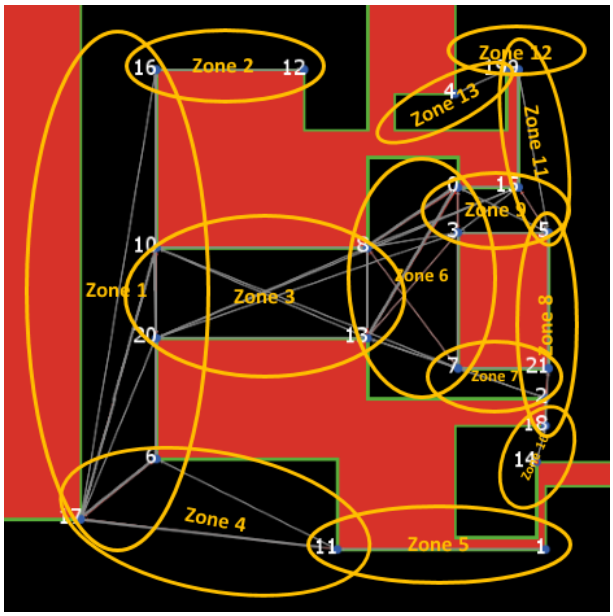


FIGURE 2.6 – Graphe de visibilité et sa décomposition en cliques

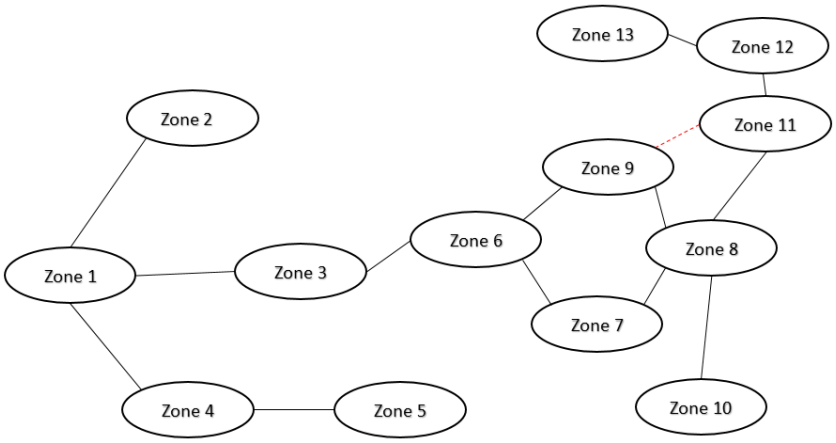


FIGURE 2.7 – Graphe de zones

Méthode de recherche naïve

Du fait que l'environnement soit structuré en chemins interconnectés dont chacun d'eux est une succession de zones, l'idée de base dans cette approche est de réaliser la décontamination d'un chemin par un agent de haut niveau considéré comme agent principal, et afin d'éviter sa recontamination, de confier la surveillance de ses zones critiques à des agents de bas niveau considérés comme des agents de garde [36]. L'opération de décontamination est considérée comme étant terminée lorsque tous les chemins de l'environ-

nement ont été visités par l'agent principal. Pour réussir l'opération de décontamination il faut donc un agent principal et autant d'agents de garde que de zones critiques dans l'environnement.

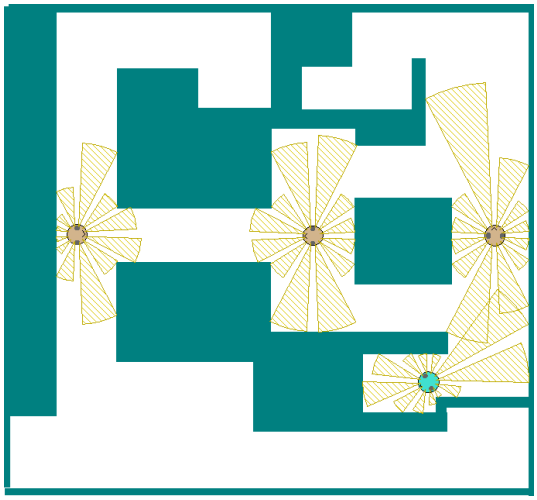


FIGURE 2.8 – Application de la méthode naïve sur un environnement.

Dans la figure 2.8, on présente le résultat de l'application de la méthode naïve sur l'environnement de la figure 2.6. L'agent principal est représenté en bleu et les trois agents de garde en marron déployés sur les zones critiques (zones 1, 6 et 8).

Méthode de recherche améliorée

Dans cette technique, la recherche démarre à partir d'une zone **feuille**, et consiste à remonter le chemin (**décontamination des zones couloirs**) jusqu'à la rencontre de la première zone **critique**. Alors un agent garde est demandé pour surveiller cette zone critique et le nombre d'agents de garde en opération est incrémenté d'une unité. La stratégie utilisée pour choisir le prochain chemin à explorer repose sur l'utilisation d'une fonction *fitness* qui calcule, sur une profondeur n, pour tous les chemins (non encore décontaminés) reliés à la zone critique courante le nombre de zones critiques présentes sur chacun d'eux. La priorité d'exploration est accordée aux chemins qui présentent le moins de zones critiques. Lorsqu'un chemin est décontaminé (visite de sa zone feuille), l'agent principal remonte jusqu'à la position de la zone critique courante surveillée par un agent garde. Lorsque tous les chemins reliés à cette zone critique courante ont été décontaminés, l'agent de garde est libéré et le nombre d'agents de garde en opération est décrémenté. L'agent principale remonte vers la précédente zone critique encore surveillée et nettoie tous les chemins qui lui sont reliés en gérant de manière dynamique le nombre d'agents de garde. Lorsque le nombre d'agents garde en opération devient égale à zéro, l'environnement est considéré comme décontaminé. Le nombre d'agents nécessaires pour décontaminer un environnement est le nombre maximal d'agents utilisés en même temps.

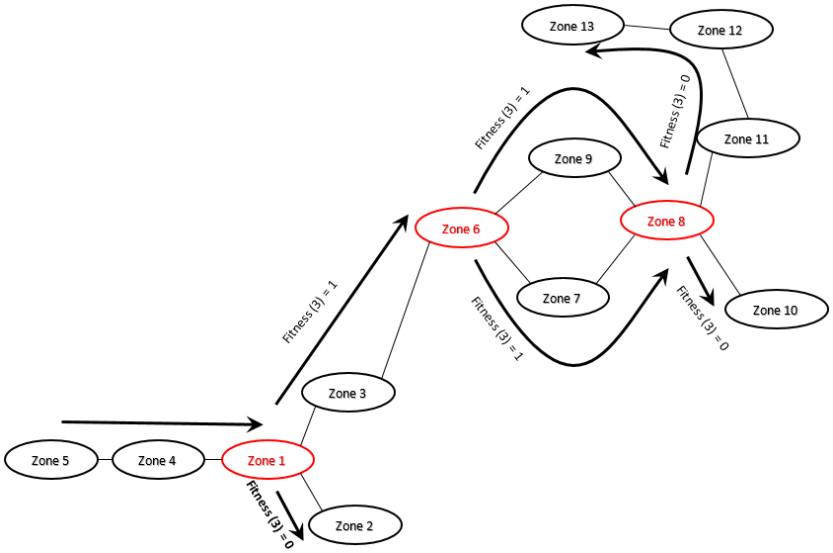


FIGURE 2.9 – Scénario de sélection de la meilleure branche avec la performance minimale

2.4.3 Recherche dans un environnement inconnu

Stratégie forward

Cette méthode [18] est inspirée du comportement humain, où la recherche doit se faire aussi tôt que possible. La priorité est accordée à la vitesse d'accès à toutes les zones de l'environnement pour minimiser le temps d'exploration au détriment de la minimisation du nombre d'agents engagés dans l'opération. L'agent principal demande un agent de garde à chaque fois qu'il découvre une nouvelle zone. C'est la stratégie de "*acting then thinking*".

L'algorithme 2.2 décrit le comportement de l'agent principal

Algorithme 2.2 Scénario de décision dans la stratégie directe.

```
if there is one Vertex in CV then
    Insert current vertex in stack
    Move to this new vertex
end if
if there is more than one vertex in CV then
    if there is no walker controlling this area (including this vertex) then
        Clone-walker
    end if
    Insert current vertex in stack
    Move to the first vertex in CV
end if
if there is no vertex in CV then
    Create tmp, the list of vertices in QL that are visible from V
    if tmp is not empty then
        if tmp contains only one vertex then
            kill-walker
        end if
        Insert current vertex in stack
        Move to this new vertex
    end if
    if tmp is empty then
        Kill-walker
        Go back
    end if
end if
```

Avec :

CV	est la liste des vertex visibles par vertex courant *V*
stack	est la liste des vertex visités, elle est utilisée pour le retour en arrière
Clone-walker	demander un agent gardien pour la zone en cours
QL	est la liste des vertex non visités
V	est le vertex en cours
Kill-walker	est de libérer un gardien (s'il existe) avec l'agent principal
Go-back	est de revenir en arrière au vertex précèdent dans la *Stack*

A titre d'exemple, avec l'environnement représenté dans la figure 2.10, et en supposant que le robot agent principal est sur le vertex 7 alors l'algorithme est dans l'état suivant :

1. vertex 7 est le vertex courant
2. *CV* contient les vertex visibles à partir du vertex courant (8, 2, 0, 3, 5, 4, 1)
3. un agent garde est demandé
4. le contenu de *CV* est transféré dans *QL*

5. vertex 7 est placé dans *stack*
6. vertex 8 devient vertex courant et le scénario se répète jusqu'au visite de tous les vertex.

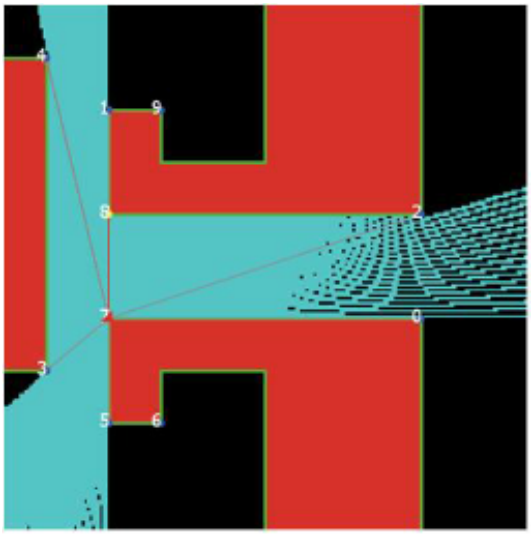


FIGURE 2.10 – Cas de vérification de visiblité

Stratégie forward-backward

Dans ce cas [17], l'agent maître visite tous les sommets de la zone courante (c'est-à-dire les sommets visibles à partir du sommet courant) et détermine s'il y a d'éventuelles zones critiques. Cette stratégie assure que les agents de garde ne sont demandés que pour des zones critiques. Cette technique est inspirée aussi par le comportement humain où, par instinct, on vérifie d'abords toute la zone en cours avant prendre une décision. C'est la stratégie "*thinking then acting*".

Pour le scénario représenté sur la figure 2.10, l'agent principal, en faisant des allers et retours entre le vertex 7 (vertex courant) et les vertex de la liste CV, visite tous les vertex de cette liste et en détermine ceux qu'on qualifie de critiques (vertex appartenant à deux cliques adjacentes). Dans ce cas, il en existe deux : les vertex (1 et 5).

La machine à états finis, représentée à la figure 2.11, modélise le comportement de l'algorithme.

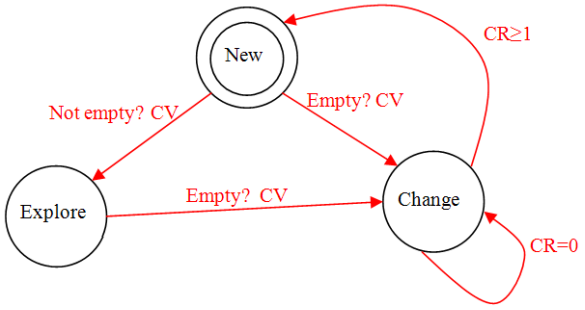


FIGURE 2.11 – Machine à état fini modélisant la stratégie forward-backward

L'algorithme est composé de trois états, dont les actions associées sont présentées ci-après :

— **New :** dans cet état, l'algorithme génère l'ensemble des vertex visibles à partir du vertex courant et établi la liste CV associée au vertex courant.
— **Explore :** cet état est utilisé pour établir la liste CR des vertex critiques à partir de la liste CV courante.
— **Change :** dans cet état l'agent principal décide de progresser dans l'exploration de l'environnement à partir d'un nouveau vertex critique (phase **forward**) et demande un agent garde pour le placer sur ce vertex, ou bien de retourner au vertex critique précédent (phase **backward**) si la zone a été totalement explorée

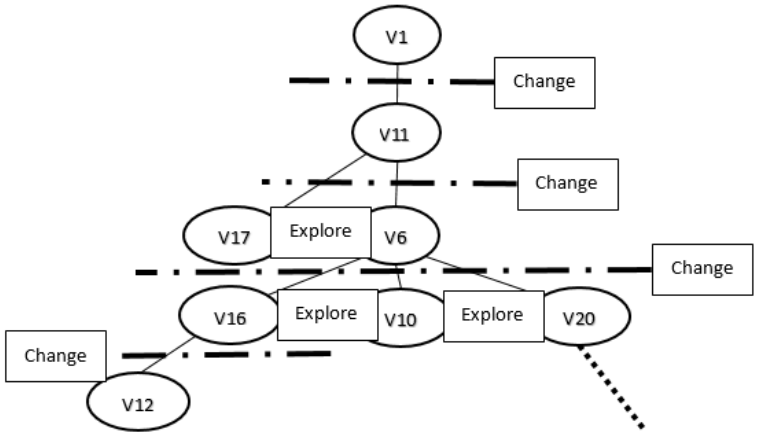


FIGURE 2.12 – Diagramme représentant l'exécution de la machine à états

Cette machine à états finis est exécutée à chaque fois que l'agent principal change de position d'un vertex à un autre.

2.5 Modélisation de l'état des zones par les réseaux de Petri

2.5.1 Introduction

Cette partie est consacrée à la modélisation et à l'analyse du graphe de zones à l'aide des réseaux de Petri. Cet outil de modélisation graphique a été utilisé pour modéliser l'évolution de l'état de décontamination des zones en fonction du nombre de robots présents dans l'environnement.

2.5.2 Modélisation de l'état des zones

Chaque zone peut être dans l'un des deux états possibles, soit l'état contaminé ou bien l'état décontaminé. Quand une zone est décontaminée, elle doit être protégée par un agent se trouvant sur la zone elle-même, ou bien par des agents se trouvant sur les chemins qui y convergent. Il est évident qu'une zone décontaminée mais non gardée peut être contaminée de nouveau.

La zone A_i a les attributs booléens : $agentA_i?$ / $guardedA_i?$ / $contamA_i?$ / $decontamAllA_i?$

L'algorithme modélisant l'évolution dans le changement d'état d'une zone est présenté dans l'algorithme 2.3.

Algorithme 2.3 l'état de la zone A_i.

La zone A_i est initialement $!guardedA_i?$, $contamA_i?$ et $!decontamAllA_i?$
if $agentA_i?$ **then**
  $guardedA_i?$
end if
if $guardedA_i?$ **then**
  $!contamA_i?$
end if
if $!agentA_i?$ **and** $\sum_j AgentA_j?$ [1] **then**
  $guardedA_i?$
end if
if $\prod_j !contamA_j?$ [2] **then**
  $decontamAllA_i?$
end if
if $decontamAllA_i?$ **then**
  $!contamA_i?$
end if

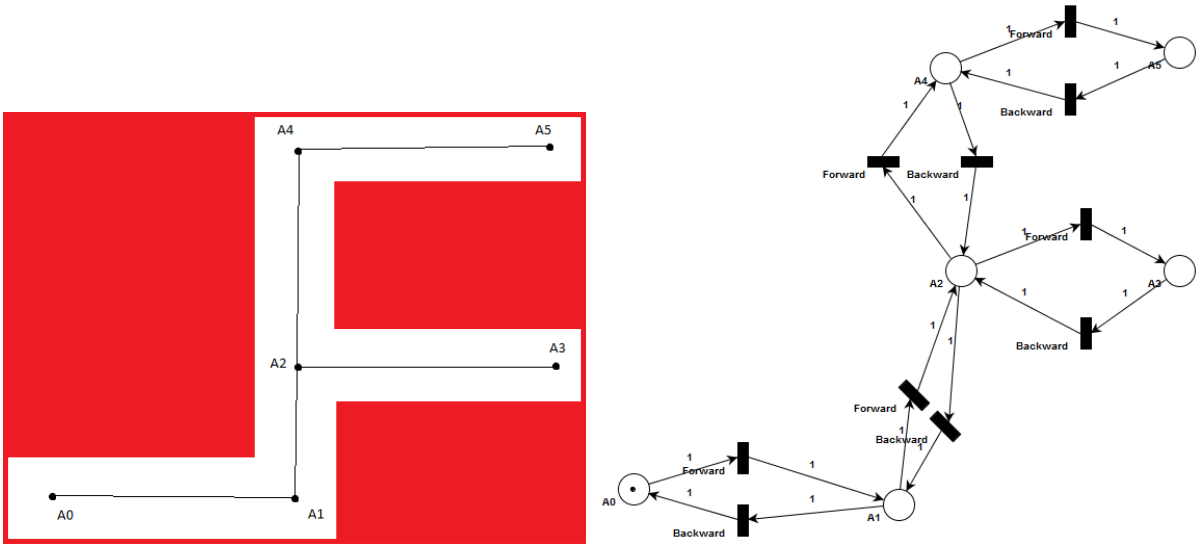
2.5.3 Conversion du graphe de zones en réseau de Petri

Afin de convertir le comportement global du système en réseau de Petri, il faut convertir chaque partie de l'algorithme en un réseau indépendant, le réseau global est une combinaison des sous-réseaux.

Réseau des zones

L'environnement global est converti en un réseau de Petri dans lequel les places représentent les zones du graphe et les transitions "incidente" et "réfléchie" correspondent aux arcs et représentent les déplacements aller et retour de l'agent principal entre les zones adjacentes (Figure 2.13).

Un jeton dans une place A_i veut dire qu'il y a un agent dans cette zone ($guardedA_i?$).



(a) Environnement avec graphe de zone (b) Réseau de Petri équivalent

FIGURE 2.13 – Exemple d'environnement avec graphe de zones et réseau de Petri équivalent.

Evolution du marquage d'une zone

Chaque place du réseau de Petri modélisant une zone du graphe est en réalité une macro place. L'évolution du marquage de cette macro-place est gérée par trois sous réseaux de Petri : le premier modélisant sa visite et sa garde par un agent, le deuxième modélisant son état contaminé-décontaminé et le dernier sous réseau de Petri modélisant la propagation de la contamination à travers les zones adjacentes

Sous réseau de Petri du comportement *guarded*?

Les deux places $guardedA_i?$ et $!guardedA_i?$ du sous réseaux de Petri de la figure 2.14 correspondent aux deux états possibles dans lesquels une zone peut se trouver. La zone est initialement non gardé (présence d'un jeton dans la place $!guardedA_i?$). Et l'arrivée d'un agent sur la zone (présence d'un jeton dans la place A_4) déclenche le franchissement de la transition $guaA_4$ entrainant un changement de l'état de la zone. Elle devient une zone gardée (présence d'un jeton dans la place $guardedA_4?$), et elle le restera tant que qu'il y a un agent présent dans la zone.

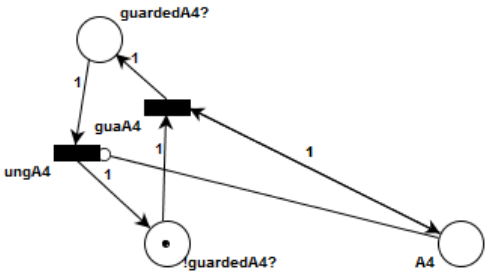


FIGURE 2.14 – Comportement *guarded*?

Sous réseau de Petri du comportement *contam*?

Les deux places $contamA_4?$ et $!contamA_4?$ correspondent aux deux états contaminé et décontaminé dans lesquels peut se trouver une zone. Initialement elle est contaminée, et elle change d'état pour devenir décontaminée dès qu'elle reçoit la visite d'un agent. Elle restera décontaminée tant que les zones qui lui sont adjacentes sont gardées ou bien décontaminées.

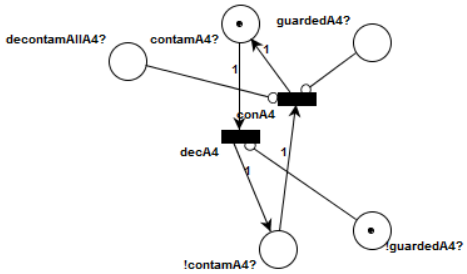


FIGURE 2.15 – Le comportement *contam*?

Sous réseau de Petri du comportement *decontamAll*?

Ce sous réseau modélise la contamination des zones adjacentes (Figure 2.16). Une zone ne peut pas être contaminée si toutes ses zones voisines sont décontaminées. Ceci est traduit par un arc inhibiteur sortant de chaque place $contamA_j?$. Si la zone est entièrement décontaminée ($decontamAllA_4?$) alors la zone reste décontaminée ($!contamA_4?$). Pour cela, un arc inhibiteur sort de la place $decontamAllA_4?$ vers la transition $conA_4$. L'état $decontamAllA_4?$ reste vraie tant que toutes les zones voisines sont décontaminées. Si une zone adjacente devient contaminée alors la place $!decontamAllA4?$ contiendra un jeton traduisant le fait que la zone A_4 peut être contaminée de nouveau. Pour cela, il y a une transition (T_{40} à T_{43}), pour chaque zone voisine A_j, possédant un arc de $decontamAllA_4?$ et $contamA_j?$ et vers $!decontamAllA_4?$ et $contamA_j?$ (voir Figure 2.16).

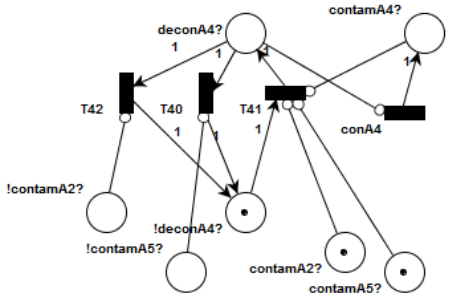


FIGURE 2.16 – Le réseau *decontamAll*?

Combinaison des comportements

En combinant les trois sous réseaux précédents modélisant l'état de chaque zone de l'environnement, illustré dans la Figure 2.13(a). on obtient le réseau de Petri de la Figure A.1 à la page 116 représentant le comportement global de l'évolution de l'état des différentes zones de cet environnement en fonction de la progression de l'agent principal et des agents de garde dans l'opération de décontamination.

2.5.4 Analyse du modèle

L'outil de modélisation PIPE

Pour analyser le modèle proposé, un outil de modélisation et d'analyse à l'aide des réseaux de Petri a été utilisé (Figure 2.17), cet outil est PIPE [3] [8]. PIPE est un outil open-source indépendant de la plateforme du système d'exploitation pour créer et analyser des réseaux de Petri de type immédiat ou stochastique, GSPNs [4] [2], PIPE est entièrement implémentée en Java pour sécuriser l'indépendance de l'outil du système d'exploitation et donner une interface graphique élégante et facile à utiliser. Cet outil graphique permet la création, la sauvegarde, le chargement et l'analyse des réseaux de Petri. PIPE a été initialement créé dans le Collège Impérial de Londres en utilisant plusieurs projets d'étudiants. Une nouvelle version avec des améliorations considérables sur différents aspects a été implémentée à l'université des Iles Baléares.

L'interface graphique de l'outil PIPE est convivial et de prise en main facile, il est composé de trois blocs : le premier bloc est la barre d'outils qui permet la saisie du graphe du réseau de Petri, le deuxième bloc contient les éléments d'analyse des propriétés du réseau de Petri, et le troisième bloc est l'espace de travail.

3. Platform-Independent Petri net Editor
4. Generalized Stochastic Petri Net(s)

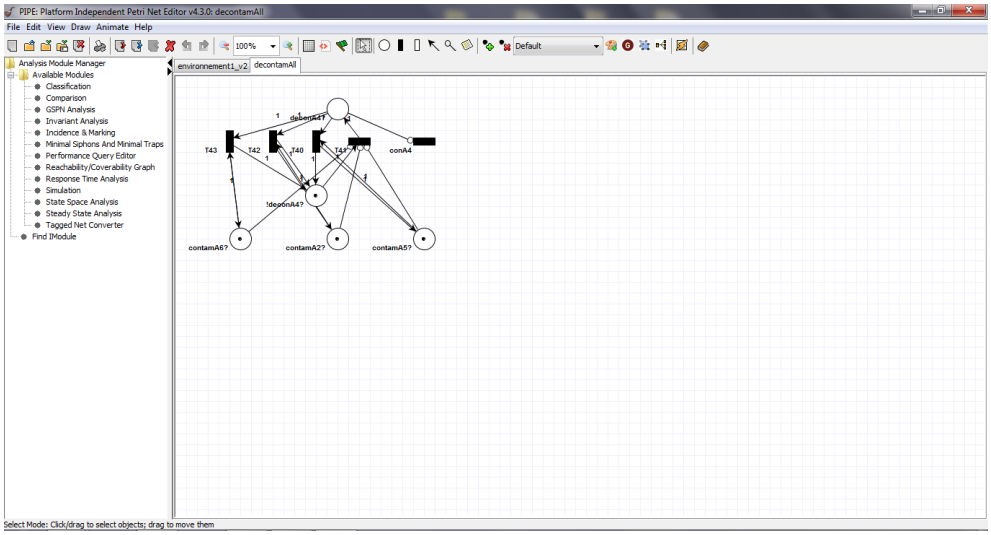


FIGURE 2.17 – Interface de l'outil PIPE

Matrice d'incidence avant

Les arcs de liaison de Transitions vers Places $Pre\,(P_j, T_j)$ peuvent être représentés dans une matrice avec $W^+\,(i,j) = Pr\,(P_i, T_j)$

Alors l'indice des lignes est la place et l'indice des colonnes est la transition. Les valeurs de la matrice sont :

$$W^+\,(i,j) = \begin{cases} 1 & s'il\ existe\ un\ arc\ de\ T_j\ vers\ P_i \\ 0 & sinon \end{cases} \tag{2.1}$$

Dans l'exemple précèdent (Figure A.1), la matrice d'incidence avant est une matrice de 42 lignes et 50 colonnes.

A titre d'exemple la table 2.3 représente une partie de la matrice d'incidence avant concernant la zone A_4.

Matrice d'incidence arrière

Les arcs de liaison de Places vers Transitions $Post\,(P_i, T_j)$ peuvent être représenté dans une matrice avec $W^-\,(i,j) = Post\,(P_i, T_j)$ Alors l'indice des lignes est la place et l'indice des colonnes est la transition. Les valeurs de la matrice sont

$$W^-\,(i,j) = \begin{cases} 1 & s'il\ existe\ un\ arc\ de\ P_i\ vers\ T_j \\ 0 & sinon \end{cases} \tag{2.2}$$

Dans l'exemple précèdent (Figure A.1), la matrice d'incidence arrière est une matrice de 42 lignes et 50 colonnes.

	F_{45}	F_{54}	F_{42}	F_{24}	$guaA_4$	$ungA_4$	$conA_4$	$decA_4$	T_{40}	T_{41}	T_{42}
A_4	0	1	0	1	1	0	0	0	0	0	0
A_2	0	0	1	0	0	0	0	0	0	0	0
A_5	1	0	0	0	0	0	0	0	0	0	0
$gA_4?$	0	0	0	0	1	0	0	0	0	0	0
$!gA_4?$	0	0	0	0	0	1	0	0	0	0	0
$cA_4?$	0	0	0	0	0	0	1	0	0	0	0
$!cA_4?$	0	0	0	0	0	0	0	1	0	0	0
$dAllA_4?$	0	0	0	0	0	0	0	0	1	0	0
$!dAllA_4?$	0	0	0	0	0	0	0	0	0	1	1
$cA_2?$	0	0	0	0	0	0	0	0	0	0	0
$!cA_2?$	0	0	0	0	0	0	0	0	0	0	0
$cA_5?$	0	0	0	0	0	0	0	0	0	0	0
$!cA_5?$	0	0	0	0	0	0	0	0	0	0	0

TABLE 2.3 – Matrice d'incidence avant

A titre d'exemple la table 2.4 représente une partie de la matrice d'incidence arrière concernant la zone A_4.

Matrice d'inhibition

Les arcs d'inhibition de Places vers Transitions $Inhib\,(P_i, T_j)$ peuvent être représenté dans une matrice avec $H\,(i,j) = Inhib\,(P_i, T_j)$ Alors l'indice des lignes est la place et l'indice des colonnes est la transition. Les valeurs de la matrice sont

$$H\,(i,j) = \begin{cases} 1 & s'il\ existe\ un\ arc\ inhibiteur\ de\ P_i\ vers\ T_j \\ 0 & sinon \end{cases} \tag{2.3}$$

Dans l'exemple précèdent (Figure A.1), la matrice d'inhibition est une matrice de 42 lignes et 50 colonnes.

A titre d'exemple la table 2.5 représente une partie de la matrice d'inhibition concernant la zone A_4

Marquage initial

On initialise la plate forme PIPE par le marquage initial sous forme d'un vecteur consigné dans la Table 2.6. Ce marquage initial peut être trouvé d'une façon intuitive :

- Initialement le(s) robot(s) est (sont) dans une des places de zones (ici place A_0).
- Toutes les zones sont initialement
  - non gardées ($!guardedA_i?$)

	F_{45}	F_{54}	F_{42}	F_{24}	$guaA_4$	$ungA_4$	$conA_4$	$decA_4$	T_{40}	T_{41}	T_{42}
A_4	1	0	1	0	1	0	0	0	0	0	0
A_2	0	0	0	1	0	0	0	0	0	0	0
A_5	0	1	0	0	0	0	0	0	0	0	0
$gA_4?$	0	0	0	0	0	1	0	0	0	0	0
$!gA_4?$	0	0	0	0	1	0	0	0	0	0	0
$cA_4?$	0	0	0	0	0	0	0	1	0	0	0
$!cA_4?$	0	0	0	0	0	0	1	0	0	0	0
$dAllA_4?$	0	0	0	0	0	0	0	0	0	1	1
$!dAllA_4?$	0	0	0	0	0	0	0	0	1	0	0
$cA_3?$	0	0	0	0	0	0	0	0	0	0	0
$!cA_3?$	0	0	0	0	0	0	0	0	0	0	0
$cA_5?$	0	0	0	0	0	0	0	0	0	0	0
$!cA_5?$	0	0	0	0	0	0	0	0	0	0	0

TABLE 2.4 – Matrice d'incidence arrière

	F_{45}	F_{54}	F_{42}	F_{24}	$guaA_4$	$ungA_4$	$conA_4$	$decA_4$	T_{40}	T_{41}	T_{42}
A_4	0	0	0	0	0	1	0	0	0	0	0
A_2	0	0	0	0	0	1	0	0	0	0	0
A_5	0	0	0	0	0	1	0	0	0	0	0
$gA_4?$	0	0	0	0	0	0	1	0	0	0	0
$!gA_4?$	0	0	0	0	0	0	0	1	0	0	0
$cA_4?$	0	0	0	0	0	0	0	0	1	0	0
$!cA_4?$	0	0	0	0	0	0	0	0	0	0	0
$dAllA_4?$	0	0	0	0	0	0	1	0	0	0	0
$!dAllA_4?$	0	0	0	0	0	0	0	0	0	0	0
$cA_2?$	0	0	0	0	0	0	0	0	1	0	0
$!cA_2?$	0	0	0	0	0	0	0	0	0	0	1
$cA_5?$	0	0	0	0	0	0	0	0	1	0	0
$!cA_5?$	0	0	0	0	0	0	0	0	0	1	0

TABLE 2.5 – Matrice d'inhibition

— contaminées ($contamA_i?$)
— non entièrement contaminées ($!decontamAllA_i?$)

	Initial			**Initial**			**Initial**
$!contamA_0?$	0		$!guardedA_2?$	1		$contamA_4?$	1
$!contamA_1?$	0		$!guardedA_3?$	1		$contamA_5?$	1
$!contamA_2?$	0		$!guardedA_4?$	1		$deconA_0?$	0
$!contamA_3?$	0		$!guardedA_5?$	1		$deconA_1?$	0
$!contamA_4?$	0		A_0	1		$deconA_2?$	0
$!contamA_5?$	0		A_1	0		$deconA_3?$	0
$!deconA_0?$	1		A_2	0		$deconA_4?$	0
$!deconA_1?$	1		A_3	0		$deconA_5?$	0
$!deconA_2?$	1		A_4	0		$guardedA_0?$	0
$!deconA_3?$	1		A_5	0		$guardedA_1?$	0
$!deconA_4?$	1		$contamA_0?$	1		$guardedA_2?$	0
$!deconA_5?$	1		$contamA_1?$	1		$guardedA_3?$	0
$!guardedA_0?$	1		$contamA_2?$	1		$guardedA_4?$	0
$!guardedA_1?$	1		$contamA_3?$	1		$guardedA_5?$	0

TABLE 2.6 – Marquage initial

Analyse d'invariance

Selon le modèle créé dans PIPE, les sous réseaux P invariants sont :

$$
\begin{aligned}
M\left(!contamA_0?\right) + M\left(contamA_0?\right) &= 1 \\
M\left(contamA_1?\right) + M\left(!contamA_1?\right) &= 1 \\
M\left(!contamA_2?\right) + M\left(contamA_2?\right) &= 1 \\
M\left(!contamA_3?\right) + M\left(contamA_3?\right) &= 1 \\
M\left(!contamA_4?\right) + M\left(contamA_4?\right) &= 1 \\
M\left(!contamA_5?\right) + M\left(contamA_5?\right) &= 1 \\
M\left(!deconA_1?\right) + M\left(deconA_1?\right) &= 1 \\
M\left(!deconA_0?\right) + M\left(deconA_0?\right) &= 1 \\
M\left(!deconA_2?\right) + M\left(deconA_2?\right) &= 1 \\
M\left(!deconA_3?\right) + M\left(deconA_3?\right) &= 1 \\
M\left(!deconA_4?\right) + M\left(deconA_4?\right) &= 1 \\
M\left(!deconA_5?\right) + M\left(deconA_5?\right) &= 1 \\
M\left(guardedA_1?\right) + M\left(!guardedA_1?\right) &= 1 \\
M\left(!guardedA_0?\right) + M\left(guardedA_0?\right) &= 1 \\
M\left(!guardedA_2?\right) + M\left(guardedA_2?\right) &= 1 \\
M\left(!guardedA_3?\right) + M\left(guardedA_3?\right) &= 1 \\
M\left(!guardedA_4?\right) + M\left(guardedA_4?\right) &= 1 \\
M\left(!guardedA_5?\right) + M\left(guardedA_5?\right) &= 1 \\
M\left(A_0\right) + M\left(A_1\right) + M\left(A_4\right) + M\left(A_5\right) + M\left(A_2\right) + M\left(A_3\right) &= 1
\end{aligned}
\tag{2.4}
$$

Discussion : En proposant les sous réseaux de Petri, une zone est soit contaminée ou non contaminée, gardée ou non gardée, entièrement décontaminée ou non. Ce qui fait que :

$$contamA_i?.!contamA_i? = 0 \tag{2.5}$$

$$guardedA_i?.!guardedA_i? = 0 \tag{2.6}$$

$$decontamAllA_i?.!decontamAllA_i? = 0 \tag{2.7}$$

Ce qui fait que les couples de places représentant les états précédents sont P-invariant avec une somme de jetons égale à 1. Alors :

$$M\left(!contamA_i?\right) + M\left(contamA_i?\right) = 1 \tag{2.8}$$

$$M\left(guardedA_i?\right) + M\left(!guardedA_i?\right) = 1 \tag{2.9}$$

$$M\left(!decontamAllA_i?\right) + M\left(decontamAllA_i?\right) = 1 \tag{2.10}$$

En plus, le réseau de Petri modélisant le graphe de zones (Figure 2.13(b)) est autonome, ce qui fait que ses jetons ne seront pas perdus. Or les jetons présentent les agents gardiens dans l'environnement. Alors à n'importe quel moment, la somme des jetons dans ce réseau est égal à n : le nombre d'agents dans l'environnement.

$$\sum_i M\left(A_i\right) = n \tag{2.11}$$

Réseau borné

Un réseau marqué est borné si toutes ses places sont bornées. Les réseaux 1-bornés sont appelés des réseaux saufs. Le réseau dans la Figure A.1 est couvert par T-invariants positifs, donc il peut être borné et vivant. En plus, Le réseau est couvert par P-invariants positifs, il est donc borné.

Taille du réseau

Dans l'exemple dans la Figure A.1, le nombre de places est 42 et le nombre de transitions est 52. En posant n le nombre de zones et nv est le nombre de voisinages, pour un environnement donné, il est possible de déduire les informations sur le nombre de places et de transitions à partir du nombre de places/transitions de chaque sous réseau.

Réseau environnement : Ce réseau présente chaque zone par une place et chaque voisine entre deux zones par deux transitions (une transition d'aller de la zone A_i vers A_j et une transition de A_j vers A_i). Alors le nombre de places et des transitions est :

$$\begin{cases} P_{env} = n \\ T_{env} = 2 \times nv \end{cases} \tag{2.12}$$

Réseau $guardedA_i?$ **:** Ce réseau représente l'état de chaque zone si elle est gardée ou pas, alors elle contient deux places, une place $guardedA_i?$ et une place $!guardedA_i?$. Il a ainsi une transition de changement d'état de $guardedA_i?$ vers $!guardedA_i?$ et le contraire. Alors le nombre de places et des transitions de ce réseau est :

$$\begin{cases} P_{guarded} = 2 \\ T_{guarded} = 2 \end{cases} \tag{2.13}$$

Réseau $contamA_i?$ **:** Ce réseau représente l'état de chaque zone si elle est contaminée ou pas, alors elle contient deux places, une place $contamA_i?$ et une place $!contamA_i?$. Il a

ainsi une transition de changement d'état de $contamA_i?$ vers $!contamA_i?$ et le contraire. Alors le nombre de places et des transitions de ce réseau est :

$$\begin{cases} P_{contam} = 2 \\ T_{contam} = 2 \end{cases} \tag{2.14}$$

Réseau $decontamAllA_i?$ **:** Ce réseau représente l'état de chaque zone si elle est entièrement décontaminée ou pas, alors elle contient deux places, une place $decontamAllA_i?$ et une place $!decontamAllA_i?$. Il a ainsi une transition de changement d'état de $!decontamAllA_i?$ vers $decontamAllA_i?$ et une transition de changement de $decontamAllA_i?$ vers $!decontamAllA_i?$ pour chaque zone voisine. Alors le nombre de places et de transitions de ce réseau est :

$$\begin{cases} P_{decontamAll} = 2 \\ T_{decontamAll} = 1 + nv_i \end{cases} \tag{2.15}$$

Avec nv_i est le nombre de voisines de la zone A_i.

Réseau global : Le nombre de places et de transitions est la somme des places et transitions des macros de chaque zone, alors :

$$\begin{cases} P = P_{env} + \sum_i P^i_{guarded} + P^i_{contam} + P^i_{decontamAll} = n + n \times 6 = 7 \times n \\ T = T_{env} + \sum_i T^i_{guarded} + T^i_{contam} + T^i_{decontamAll} = 2 \times nv + n \times 5 + \sum_i nv_i \end{cases} \tag{2.16}$$

Or $\sum_i nv_i = 2 \times nv$ (car chaque zone A_i voisine à A_j alors A_j est voisine à A_i) alors

$$\begin{cases} P = 7 \times n \\ T = 5 \times n + 4 \times nv \end{cases} \tag{2.17}$$

En appliquant le cas de l'environnement dans la Figure 2.13(b) où $n = 6$ et $nv = 5$ alors $P = 42$ et $T = 50$, ce qui vérifie l'implémentation sur PIPE.

Graphe de marquage

Le graphe des marquages (R, M_0) est un graphe orienté dont les nœuds sont les marquages de $A\,(R, M_0)$, et chaque arc relie un marquage à un autre, obtenu par le franchissement d'une transition t_{ij} : alors on a $M_i \overset{t_{ij}}{\rightarrow} M_j$.

Le réseau de Petri représentant le comportement de l'environnement est toujours borné, alors il a toujours un graphe de marquage fini. Dans le cas de la figure A.1, le

graphe contient 469 états et 669 arcs. Il est évident que sa représentation graphique reste difficile.

Néanmoins, pour n'importe quelle méthode de décontamination, les marquages des places A_i et $contamA_i?$ peuvent être représentés dans une matrice à deux colonnes ; Le marquage initial des places A_i est donné par le vecteur colonne de gauche, dont tous les éléments sont nuls, sauf un et sa valeur indique le nombre d'agents robots nécessaires à la décontamination. Le marquage initial des places $contamA_i?$ est placé dans le vecteur colonne de droite et il est égal au vecteur unité. Le marquage final est atteint, et le nombre d'agents robots mobilisés pour l'opération de décontamination est validé, lorsque le vecteur colonne relatif aux places $contamA_i?$ devient égal au vecteur nul.

$$M_0 = \begin{bmatrix} A_0 & contamA_0 \\ A_1 & contamA_1 \\ A_2 & contamA_2 \\ A_3 & contamA_3 \\ A_4 & contamA_4 \\ A_5 & contamA_5 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \overset{\sigma_1}{\rightarrow} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 2 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \overset{\sigma_2}{\rightarrow} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} = M_{final} \tag{2.18}$$

Avec :

— $\sigma_1 = 2 \times Forward_{01} \rightarrow 2 \times Forward_{12}$
— $\sigma_2 = 1 \times Forward_{23/32} \rightarrow 1 \times Forward_{24} \rightarrow 1 \times Forward_{45}$

2.6 Conclusion

Ce chapitre explique l'utilisation des graphes pour la modélisation de l'environnement ainsi que les méthodes de recherche dans un environnement connu et inconnu ou partiellement connu. Par suite, un modèle par les réseaux de Petri a été proposé pour pouvoir valider les techniques de recherche par les graphes.

Le chapitre suivant présente l'application de la poursuite-évasion sur les systèmes multi-agent.

Chapitre 3

Planification de trajectoires des agents mobiles

3.1 Introduction

Pour permettre à l'agent principal de visiter tous les vertex d'une zone afin d'établir la liste CV et d'en extraire ceux qui sont critiques, il faut planifier ses déplacements en établissant des trajectoires sûres et rapides. Deux techniques basées sur la génération de champs de potentiel [34], permettent de réaliser des cartes de potentiel, dont chacune des cellules, disponibles pour la navigation, est affectée d'un couple de valeurs, la première valeur mesurant sa proximité de la cellule cible (vertex à visiter) et la seconde traduisant son éloignement des obstacles (éloignement des murs). Des méthodes de synthèse, combinant ces deux cartes de potentiel, permettant d'établir une trajectoire à la fois sûre et relativement rapide, ont été développées.

3.2 Hypothèses de travail

Les environnements retenus dans notre travail sont, pour la partie simulation, des cartes d'édifices publics enregistrés sous forme d'images, et pour la partie réalisation pratique, des environnements sous forme de labyrinthe, développés en tenant compte de la taille des robots et de l'espace disponible. L'acquisition des images de l'environnement pratique est réalisé à l'aide d'une caméra de type Webcam, de résolution 1024×768 pixels, à partir d'une vidéo VGA.

Les hypothèses suivantes ont été retenues pour la modélisation de notre environnement :

— L'environnement dans lequel évolue le robot est divisé en petites cellules carrées ayant une taille fixe en relation directe avec la taille du robot.
— Le centre de chaque cellule est localisé par deux coordonnées x et y.

— L'environnement est structuré en deux sous-espaces l'un contenant les cellules libres et disponibles pour la navigation, l'autre contenant les cellules occupées par les obstacles et donc indisponibles pour la navigation.
— Une cellule occupée partiellement est considérée comme totalement occupée et une cellule exempte de tout obstacle est considérée comme libre.
— Deux types de voisinages sont possibles :

(a) Connectivité 4 directions

(b) Connectivité 8 directions

FIGURE 3.1 – Directions possibles

Le premier type de voisinage (quatre voisins) est utilisé pour établir les cartes des potentiels et le deuxième (huit voisins) pour établir la carte de navigation.
— La trajectoire optimale proposée tient compte des aspects géométriques des mouvements et néglige les aspects dynamiques et de contrôle (Figure 3.2).

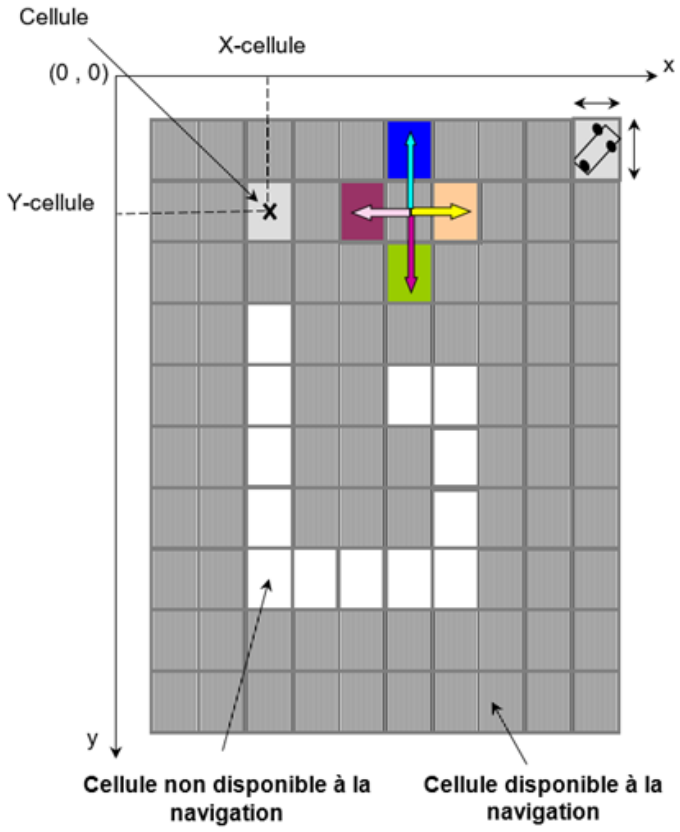


FIGURE 3.2 – Décomposition de l'environnement en cellules

3.3 Cartes de potentiel d'un environnement

3.3.1 Carte des chemins les plus sûrs

Algorithme de propagation du potentiel

L'idée de base de cet algorithme est de trouver le chemin le plus sûr et donc le chemin le plus éloigné des obstacles. Ceci est réalisé en affectant un potentiel négatif aux cellules frontières, un potentiel positif très grand aux cellules disponibles à la navigation et un potentiel nul à l'ensemble des cellules voisines des cellules frontières (Figure 3.3).

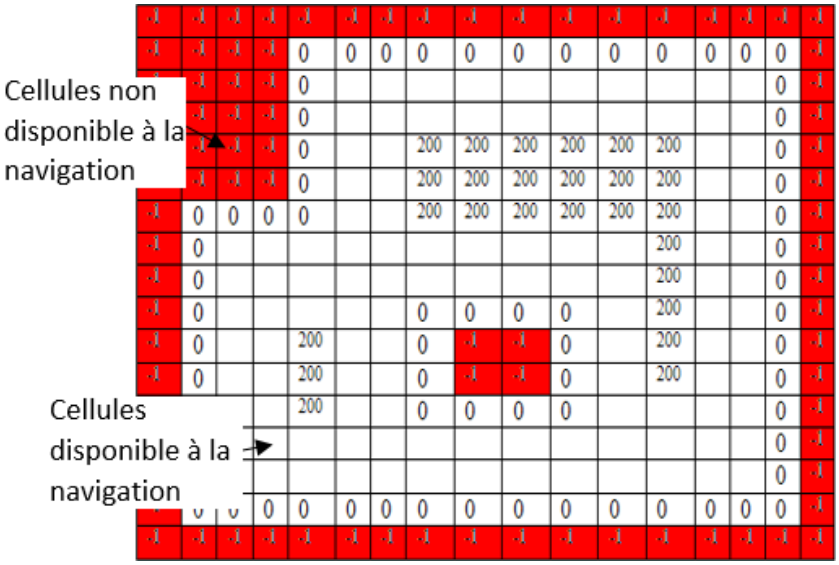


FIGURE 3.3 – Affectation de potentiel aux cellules

La technique de propagation du potentiel utilisée dans ce travail opère de la manière suivante : En partant des cellules non disponible à la navigation et se trouvant à la frontière, le phénomène de diffusion consiste à incrémenter de 1 le potentiel des cellules voisines (Figure 3.4).

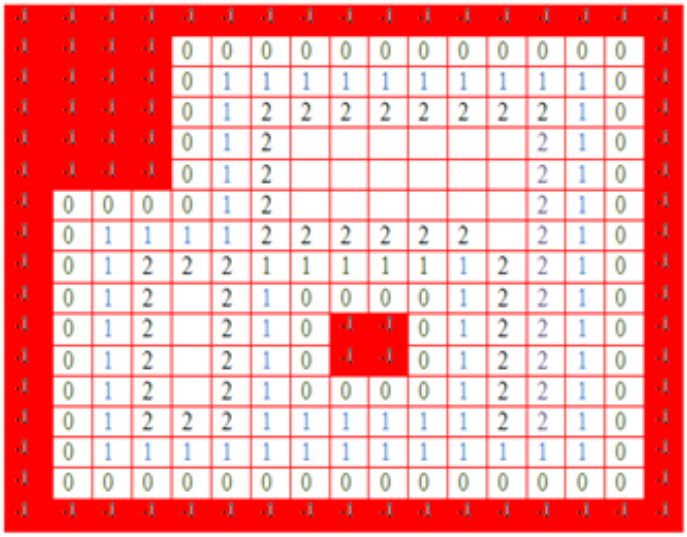


FIGURE 3.4 – Diffusion de potentiel

A leur tour ces cellules, dont le potentiel a été modifié, sont autorisées à diffuser. Ce phénomène continue d'agir par vagues successives (wave propagation) jusqu'à l'obtention de la situation finale représentée par la figure 3.5.

Les cellules disponibles à la navigation vont avoir des valeurs de potentiels positives, en relation directe avec leur éloignement des obstacles. Celles qui sont les plus éloignées des obstacles vont avoir des valeurs très grandes, et celles qui sont proches des obstacles vont avoir des valeurs proches de zéro.

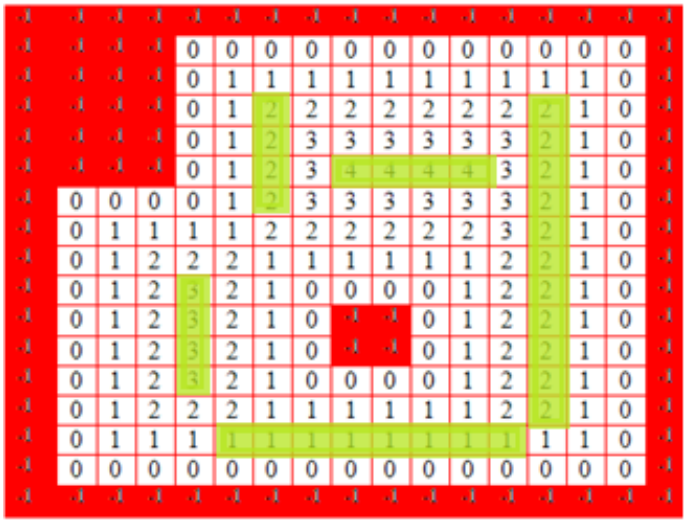


FIGURE 3.5 – Diffusion finale de potentiel et établissement de tronçons de sécurité.

Soit $Ps\,(cellule\,(i,j)) = k$, k représente la valeur du potentiel de la cellule(i,j) et correspond à la plus petite distance entre la cellule courante et la cellule frontière la plus proche. La distance utilisée dans cette technique de diffusion n'est pas une distance euclidienne, elle tient compte de la nature de la connectivité entre les quatre cellules

voisines.

L'algorithme fonctionne de la manière suivante :

En partant de l'ensemble W_0 contenant les cellules disponibles à la navigation et voisines des cellules frontières, portées au potentiel zéro, on crée l'ensemble W_1 contenant les cellules voisines des cellules de l'ensemble W_0 et dont le potentiel a été incrémenté de 1.

A leur tour, les cellules de l'ensemble W_i dont le potentiel est de niveau i vont participer à la création de l'ensemble W_{i+1} dont le potentiel est de niveau $i+1$.

Cette opération se poursuit tant que le nouvel ensemble crée est différent de l'ensemble vide

Ces opérations sont résumées dans les pseudo-codes suivants :

Algorithme 3.1 Propagation de potentiel

Initialiser W_0 = ensemble des cellules voisines des cellules frontières ; $i = 0$
Initialiser $Wi + 1 = \phi$
for all cellule $\in W_i$ **do**
  chercher les quatres cellules voisines, incrémenter leur potentiel de 1 et les insérer dans W_{i+1}.
end for
if $W_{i+1} = \phi$ **then**
  terminer SINON $i = i + 1$ et aller à l'étape 2.
end if

détermination des tronçons de chemins les plus sûrs

Les chemins les plus sûrs, utilisés pour la navigation, sont modélisés par des tronçons rectilignes, correspondant aux lignes de crête. Chaque tronçon est encadré par deux cellules nœuds (Figure 3.6).

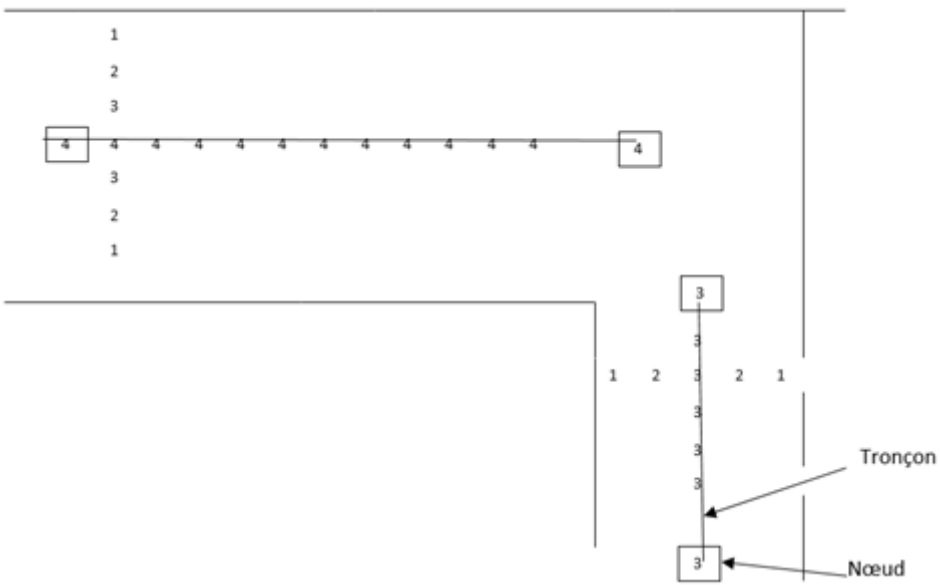


FIGURE 3.6 – Dessin de tronçon horizontal et vertical

Un tronçon horizontal (selon la ligne i) est obtenu en reliant les cellules dont les valeurs de potentiel sont des maximums locaux verticaux (selon les colonnes) et de même valeur. Le maximum local selon la colonne j vérifie l'une des relations suivantes selon la largeur du couloir :

$$\begin{cases} Ps\,(i, j-1) < Ps\,(i, j) > Ps\,(i, j+1) \\ Ps\,(i, j-1) < Ps\,(i, j) = Ps\,(i, j+1) > Ps\,(i, j+2) \end{cases} \tag{3.1}$$

Le tronçon de chemin horizontal est obtenu en faisant la concaténation des cellules voisines se trouvant sur une de crête et ayant la même valeur de potentiel.

De même pour un tronçon vertical (selon la colonne j) est obtenu en reliant les cellules dont les valeurs de potentiel sont des maximums locaux horizontaux (selon les lignes) et de même valeur. Le maximum local selon la ligne i vérifie l'une des relations suivantes selon la largeur du couloir :

$$\begin{cases} Ps\,(i-1, j) < Ps\,(i, j) > Ps\,(i+1, j) \\ Ps\,(i-1, j) < Ps\,(i, j) = Ps\,(i+1, j) > Ps\,(i+2, j) \end{cases} \tag{3.2}$$

Le tronçon de chemin vertical est obtenu en faisant la concaténation des cellules voisines se trouvant sur une de crête et ayant la même valeur de potentiel.

Un tronçon est une voie de communication sans croisement. La trajectoire optimale est obtenue en raccordant les différents tronçons, se trouvant entre la position courante du robot et le vertex cible à visiter, par des rajouts d'arcs correspondant à des changements de couloir.

3.3.2 Carte des chemins les plus rapides

Algorithme de propagation du potentiel

L'idée de base de cet algorithme est de trouver le chemin le plus rapide vers la cible. Ceci est réalisé en affectant un potentiel négatif aux cellules frontières (non disponibles à la navigation), un potentiel positif très grand aux cellules disponibles à la navigation et un potentiel égal à zéro à la cellule cible (vertex à visiter).

$Pd\,(cellule\,(i,j)) = k$, k représente la distance entre la cellule courante et la cellule cible calculée par une technique de diffusion de potentiel à partir de la cellule cible En partant de l'ensemble W_0 contenant la cellule cible portée au potentiel zéro on crée l'ensemble W_1contenant les quatre cellules voisines (connectivité 4 voisins) de la cellule cible et dont le potentiel a été incrémenté de 1.

A la i^{e} itération les cellules de l'ensemble W_i dont le potentiel est de niveau i vont participer à la création de l'ensemble W_{i+1} dont le potentiel est de niveau $i+1$.

Cette opération se poursuit tant que le nouvel ensemble crée est différent de l'ensemble vide

Ces opérations sont résumées dans l'algorithme suivant :

Algorithme 3.2 Propagation de front de vague.

Initialiser $W_0 = cellule\ cible\,;\ i = 0$
Initialiser $Wi + 1 = \phi$
for all cellule $\in W_i$ **do**
  chercher les quatre cellules voisines disponibles pour la navigation, incrémenter leur potentiel de 1 et les insérer dans W_{i+1}.
end for
if $W_{i+1} = \phi$ **then**
  terminer SINON $i = i + 1$ et aller à l'étape 2.
end if

Les valeurs affichées sur la figure suivante (Figure 3.7) représentent les distances de la cible aux frontières.

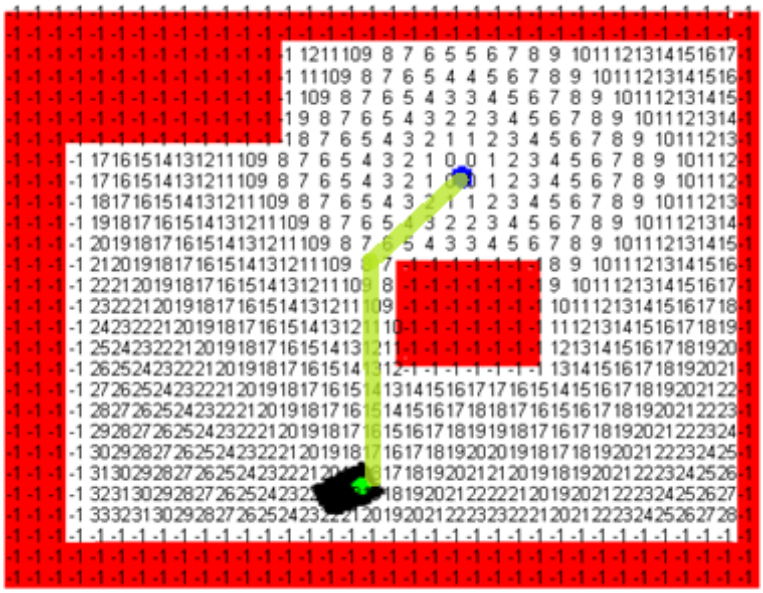


FIGURE 3.7 – Résultat de la diffusion du potentiel à partir de la cellule cible en bleu

Établissement de trajectoire le plus rapide

La trajectoire la plus rapide est établie en réalisant une descente de potentiel, à partir de la cellule correspondant à la position courante du robot vers la cellule cible portée au potentiel zéro, en utilisant la notion des 8-voisins. La force majeur de cette méthode est d'avoir un seul minimum, ce qui permet d'éviter les blocages sur les optimums locaux, malheureusement la trajectoire obtenue, a tendance à s'approcher trop des obstacles, ce qui constitue un inconvénient pour la sécurité des déplacements.

3.3.3 Synthèse de trajectoire à partir des deux cartes de potentiels

Introduction

La carte de potentiel du chemin le plus rapide, en présentant un seul minimum global correspondant au potentiel de la cellule cible, évite les pièges liés au blocage du robot sur les minimums locaux. Malheureusement la trajectoire proposée a tendance à s'approcher trop des murs des couloirs traversés. La carte de potentiel du chemin le plus sûr propose des chemins qui s'éloignent, effectivement, des obstacles, mais présente l'inconvénient d'avoir plusieurs maximums locaux. Des techniques de synthèse de trajectoires à partir de la fusion des données issues des deux cartes permettent d'établir des trajectoires robustes (absence de minimum local) à la fois sûre et relativement rapide.

Direction des déplacements

Du fait que les environnements soient de type labyrinthes constitués de couloirs et d'intersections perpendiculaires, l'évolution du robot se fait selon l'une des huit directions représentées sur la figure 3.8, appelée directions de Freeman [25] qui correspondent aux cellules voisines de la cellule occupée par le robot.

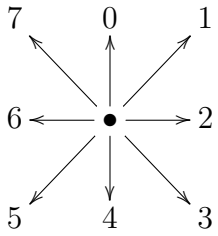


FIGURE 3.8 – Les 8 directions de Freeman

La trajectoire du robot sera constituée de déplacements successifs suivant à chaque fois l'une des directions de Freeman.

A partir de la cellule correspondante à la position courante du robot, on établit 2 matrices de taille 3×3 centrées sur cette dernière. La première matrice intègre les informations relatives au potentiel de descente (Pd) des huit cellules voisines de la cellule occupée par le robot et montre les directions de déplacement candidates pour évoluer vers la cellule cible, alors que la seconde matrice intègre les informations relatives au potentiel de sécurité (Ps) et affiche les directions candidates pour atteindre la ligne de crête la plus proche.

Dans ce cas le robot va être confronté à deux types de forces, la première exercée par le potentiel Pd, attractive vers la cible, et la seconde exercée par le potentiel Ps, répulsive du danger (frontières ou murs), phénomène de *runaround* d'Isaac Asimov [4].

Les éléments de la matrice de pondération associée aux directions candidates au déplacement sont calculés par l'expression heuristique donnée dans l'équattion 3.3 :

$$P(n,m) = Ps_1 \times ((Pd_0 - Pd_1) + (Ps_1 - Ps_0)) \tag{3.3}$$

Où Pd_0 et Ps_0 représentent respectivement les deux potentiels (descente et sécurité) de la position actuelle du robot et le Pd_1 et Ps_1 représentent respectivement les deux potentiels (descente et sécurité) de la cellule (n,m) voisine de la cellule actuelle.

La trajectoire du robot peut être décomposée en Cinq phases :

— **Phase1 :** recherche de la ligne de crête de plus grande valeur.
— **Phase2 :** évolution selon la descente de potentiel sur une ligne de valeur maximale.
— **Phase3 :** changement de couloir avec recherche de la ligne de crête maximale.

— **Phase4 :** tabuler entre phase 2 et 3.
— **Phase5 :** descente du gradient vers la cible et arrêt.

Le graphe suivant représente l'évolution du robot entre les quatre phases :

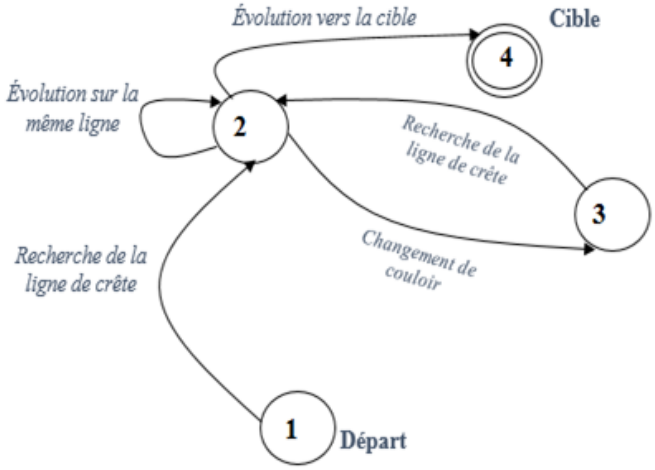
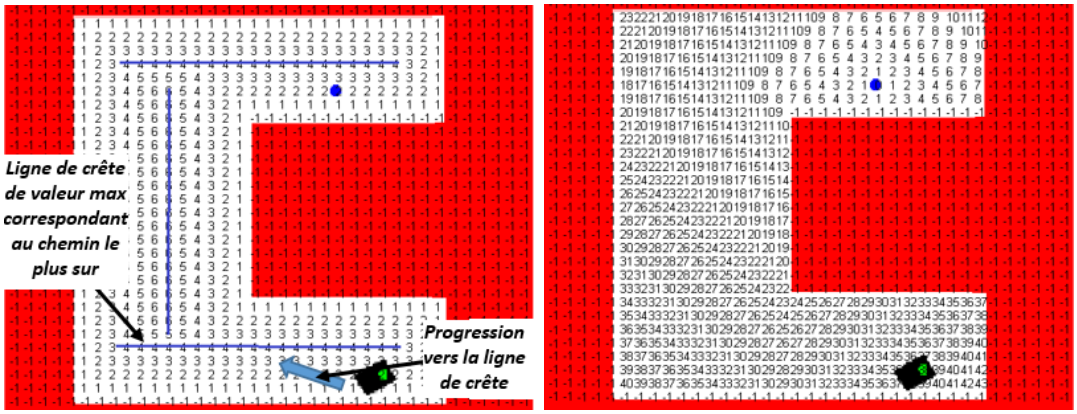


FIGURE 3.9 – Représentation de l'évolution du robot par le graphe d'état

La détermination des directions candidates aux déplacements est illustrée, sur l'environnement présenté sur les figure 3.10, pour les phases 1, 2, et 3.

Phase 1 : recherche de la ligne de crête.



(a) Carte de Ps (b) Carte de Pd

FIGURE 3.10 – Recherche de la ligne de crête

Les deux matrices de potentiel Ps et Pd relatives à la position actuelle du robot permettent d'établir les éléments de la matrice de pondération des directions candidates au déplacement.

Les directions de déplacement pondérées avec des coefficients positifs sont des directions acceptables et celle qui a le plus grand coefficient positif est la meilleure candidate.

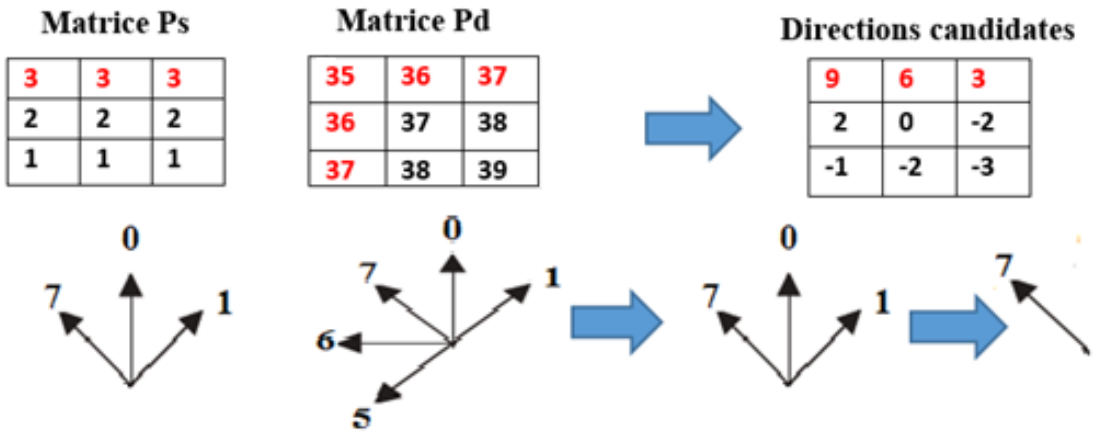
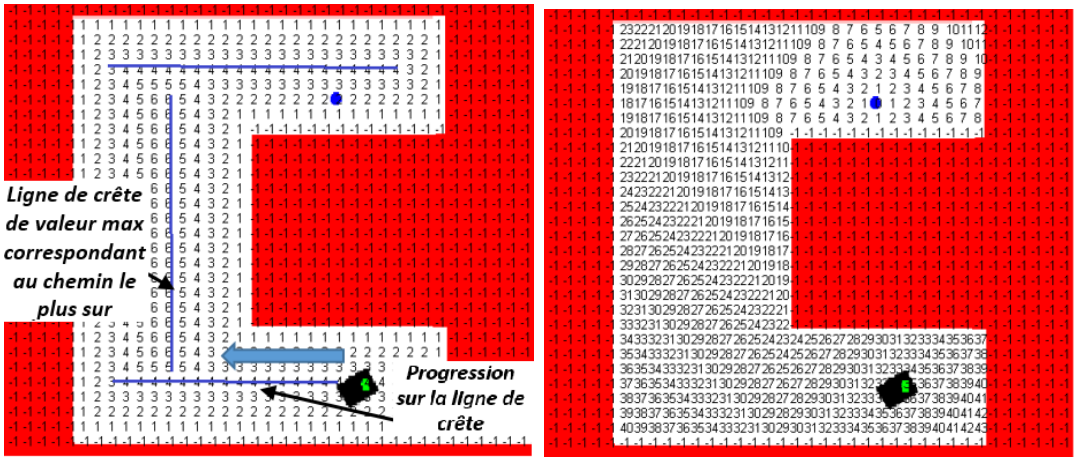


FIGURE 3.11 – Choix de la direction 7

La direction 7 permet d'accéder rapidement à une ligne de crête ou la sécurité est maximale, c'est la direction retenue pour le déplacement.

Phase 2 : Le robot se trouve au milieu du couloir sur la ligne de crête maximale.



(a) Carte de Ps (b) Carte de Pd

FIGURE 3.12 – Progression sur la ligne de crête

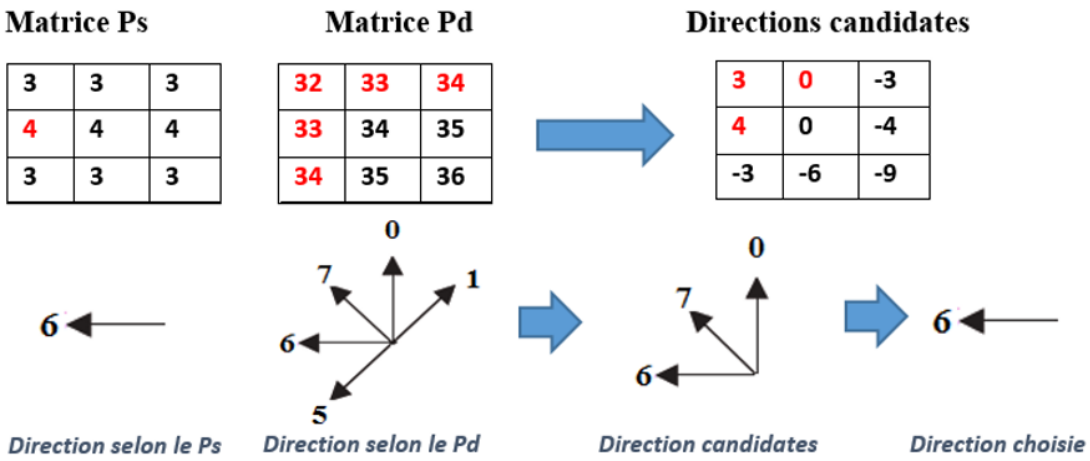
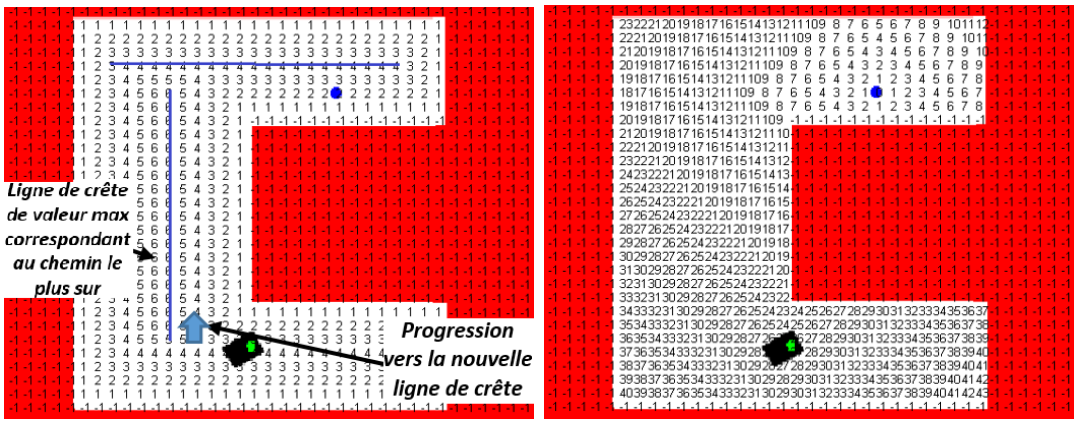


FIGURE 3.13 – Choix de la direction 6

La direction 6 réalise une descente du gradient sur une ligne de crête de valeur maximale, cette situation correspond à des conditions idéales de déplacement.

Phase 3 : Changement de couloir, cette situation correspond à la recherche de la nouvelle ligne de crête de valeur maximale (Phases 1 et 2), et progression vers la cellule destination : descente du gradient.



(a) Carte de Ps (b) Carte de Pd

FIGURE 3.14 – Progression sur la ligne de crête

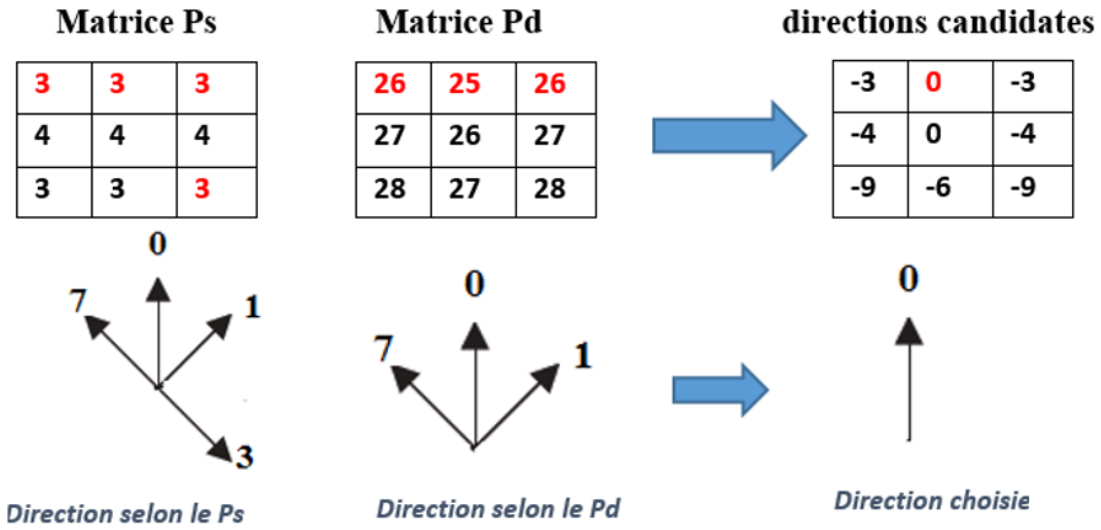


FIGURE 3.15 – Choix de la direction 0

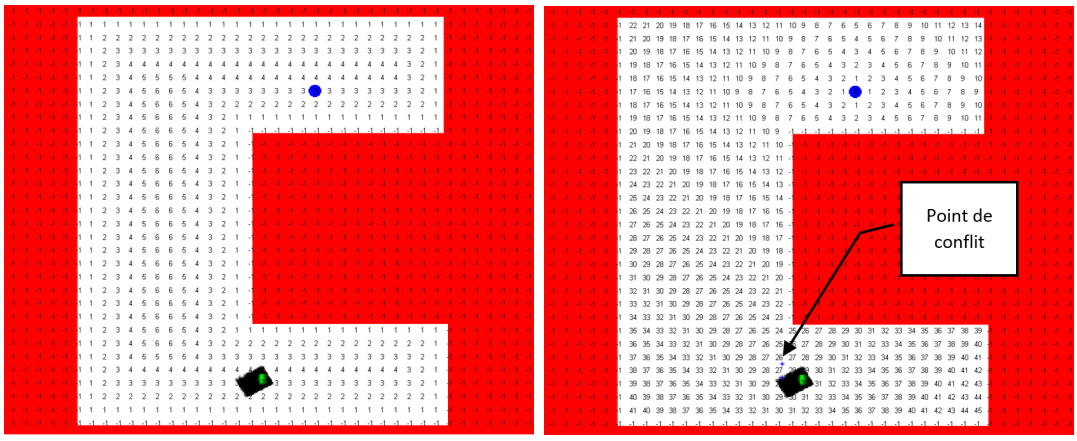
La direction 0 indique l'entame du changement de couloir.

Du fait que les couloirs ont des dimensions différentes, une variation du potentiel de sécurité accompagne souvent la recherche de la ligne de crête du nouveau couloir.

Lorsque le robot a atteint la ligne de crête du nouveau couloir, la navigation continue normalement selon l'une des trois situations vues précédemment.

Au cours d'un changement de couloir, un problème de minimal local relatif au potentiel de descente peut apparaitre et générer un conflit dans le choix de la direction d'évolution. Trois techniques de gestion de conflit ont été élaborées.

Gestion de conflits



(a) Carte de *Ps* (b) Carte de *Pd*

FIGURE 3.16 – Situation de conflit.

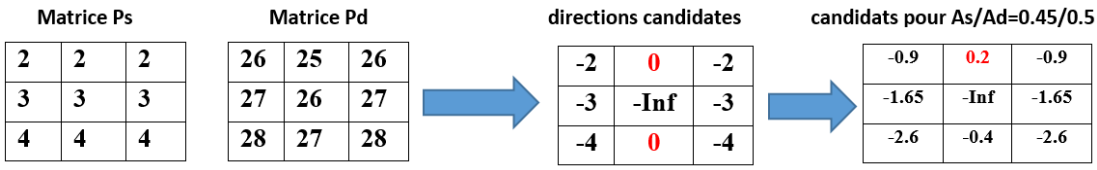


FIGURE 3.17 – Situation de conflit gérée par la pondération.

Lorsque deux directions de même pondération sont proposées pour la navigation (Figure 3.16), certaines directions sont associées à la détection précoce d'un changement de couloir, donc à la détection de la présence d'un vertex, alors que les autres sont associées à la navigation en toute sécurité sur le couloir actuel, la gestion de ce conflit est gérée par le système de relations 3.4 :

$$P(n,m) = Ps_1 \times (Ad \times (Pd_0 - Pd_1) + As \times (Ps_1 - Ps_0)) \tag{3.4}$$

Soumise aux contraintes :

$$\begin{cases} Ad + As = 1 \\ As < Ad \end{cases} \tag{3.5}$$

La direction proposée favorise la vitesse de progression vers la cellule cible.

Les deux paramètres Ad et As sont associés respectivement à la vitesse et à la sécurité des déplacements.

Génération de trajectoires

Trois techniques de génération de trajectoires, basées sur la synthèse des cartes de potentiel, sont proposées :

— **La première approche,** appelée technique de déplacement en mode normal, consiste à déterminer les directions de déplacement en appliquant la relation 3.3 et à faire appel au système de relation 3.4 en cas de conflit. L'algorithme 3.3 résume la génération de trajectoire par cette technique de synthèse.

Algorithme 3.3 Algorithme de navigation par la technique de déplacement en mode normal.

```
cellule courante := cellule de départ
while si cellule courante est différente de cellule cible do
  détermination de la direction des déplacements par la relation 3.3
  if présence de conflit then
    appliquer relation 3.4
  end if
  cellule courante := cellule déterminée
  ajouter la cellule courante à la liste des cellules constituant la trajectoire
end while
```

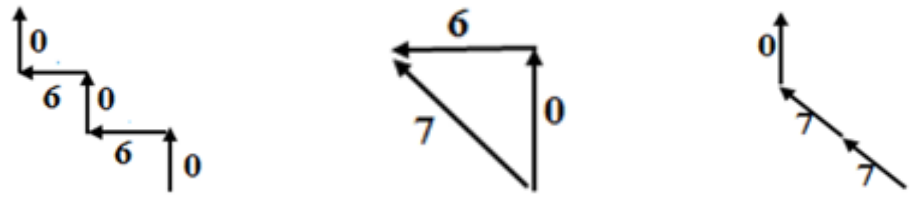
Les trajectoires obtenues sont assez irrégulières et sont peu respectueuses de la dynamique du robot.

— **La deuxième approche,** appelée technique de déplacement en mode pondéré, consiste à appliquer en permanence le système de relation 3.4.
Les directions favorisant la vitesse de progression vers la cellule cible sont privilégiées.
Les trajectoires obtenues sont plus régulières et plus rapides que les précédentes.

— **La troisième approche,** appelée technique de déplacement en mode mixe ou combiné, consiste à calculer en permanence les directions de déplacement proposées par les deux méthodes présentées ci-dessus, et privilégier les directions de déplacement sur les couloir à fort potentiel de sécurité lorsque la différence du potentiel de descente entre les cellules proposées est inférieur à un certain seuil, et donner la priorité aux directions favorisant la descente rapide vers la cellule cible lorsque ce seuil est atteint ou dépassé.
Les trajectoires obtenues sont plus douces pour la navigation et gèrent au mieux les changements de couloirs.

3.3.4 Lissage de la trajectoire

Lorsque la trajectoire comporte des variations de directions sur des courtes distances telles que celles présentées dans la figure 3.18, correspondant à des changements de direction de 90°, une opération de lissage de la trajectoire, respectant mieux la dynamique du robot, est réalisée. Elle consiste à faire au robot des déplacements avec des directions de valeur moyenne 45°. Sur l'exemple de la figure 3.18, le bout de trajet, exprimé dans l'alphabet de Freeman donne la phrase : 06060, qui peut être transformée en une phrase plus courte (donc plus rapide) et plus respectueuse de la dynamique du robot : 770



(a) Chemin avant lissage

(b) Modification requise

(c) Chemin après lissage

FIGURE 3.18 – Exemple de lissage d'une trajectoire

3.4 Exemples d'établissement de trajectoires

3.4.1 Introduction

Dans une première phase on présente deux environnements (figures 3.19) pour tester les algorithmes de génération de trajectoires développés dans la partie précédente.

La navigation, pour le premier environnement, dans deux couloirs très larges reliés par un passage relativement étroit, met en évidence la présence d'un conflit issu de la proposition de deux directions, l'une proposant de continuer de naviguer sur la crête de valeur de sécurité maximale et de ne pas quitter le couloir actuel, alors que la deuxième direction propose de s'engager dans le couloir étroit pour aller rapidement vers la cellule cible, quitte à naviguer sur une ligne de sécurité moindre.

Le deuxième environnement, de part la position courante du robot et la position de la cellule cible, les deux forces en présence pour générer des directions sont presque en permanence en conflit.

Dans une deuxième phase on présente neuf (09) environnements extraits à partir de la base de donnés et correspondant aux environnements réels dans lesquels une société d'agents mobiles peut évoluer pour des missions de neutralisation d'intrus ou porter assistance à des personnes en danger.

Pour pouvoir comparer deux trajectoires, on introduit un critère appelé ratio de sécurité, noté RS, et qui permet de mesurer la qualité d'une trajectoire. Ce critère est établi de la manière suivante :

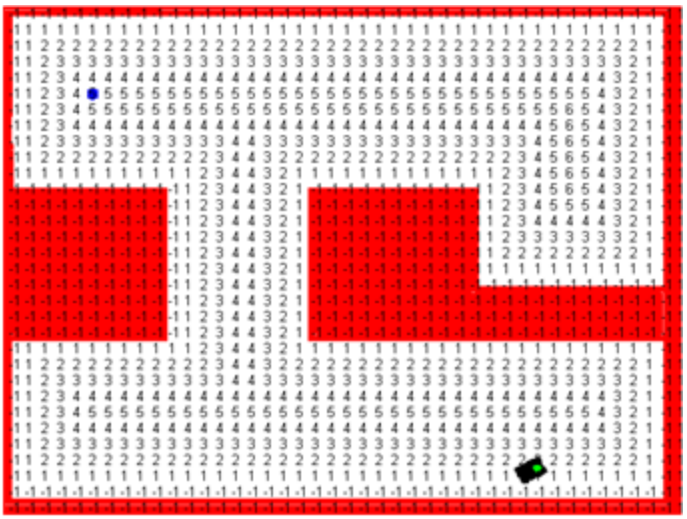
$$RS = Ps_{Total}/nb_{cellules} \tag{3.6}$$

Où

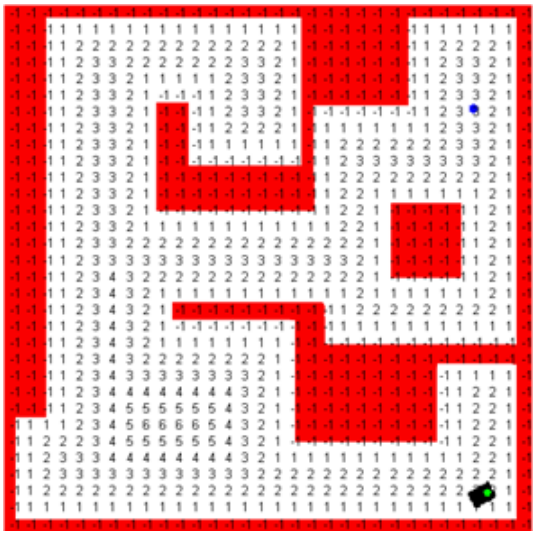
$$Ps_{Total} = \sum_{i=1}^{nb_{cellules}} Ps\,(i) \tag{3.7}$$

et $nb_{cellules}$ représente le nombre totale de cellules présente dans la trajectoire et $Ps\,(i)$ représente le potentiel de sécurité de la i^{e} cellule comptée à partir de la position courante de l'agent mobile.

Le ratio de sécurité (RS), obtenu en divisant le Ps_{Total} par le $nb_{cellules}$, permet de mesurer la qualité d'une trajectoire en déterminant le niveau moyen de sécurité associé aux cellules constituant la trajectoire.



(a) Potentiel de sécurité pour l'environnement 1

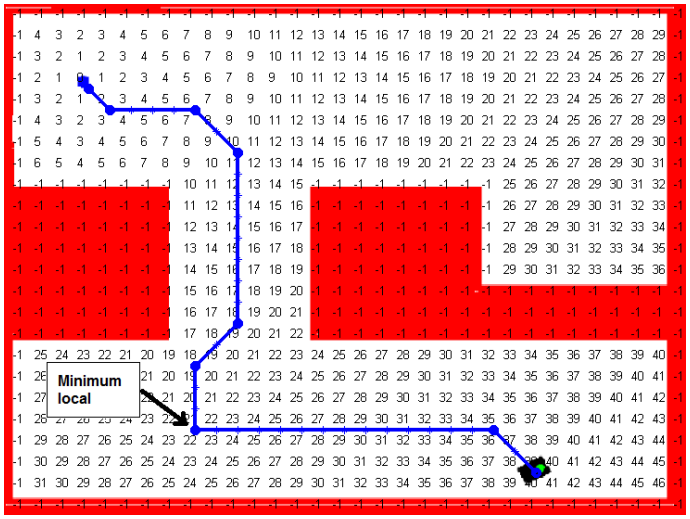


(b) Potentiel de sécurité pour l'environnement 2

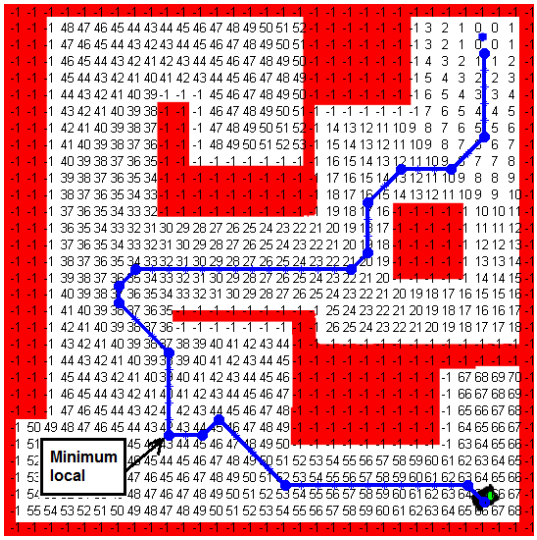
FIGURE 3.19 – Affectation du potentiel de sécurité

Technique de génération de trajectoire en mode normale

En l'absence de conflits, les directions de déplacements sont obtenues en appliquant la relation 3.3 et en présence de conflits, la direction de déplacement est obtenue en application le système de relations 3.4.



(a) Chemin pour l'environnement 1



(b) Chemin pour l'environnement 2

FIGURE 3.20 – Cartographie de l'environnement par la méthode de Pondération

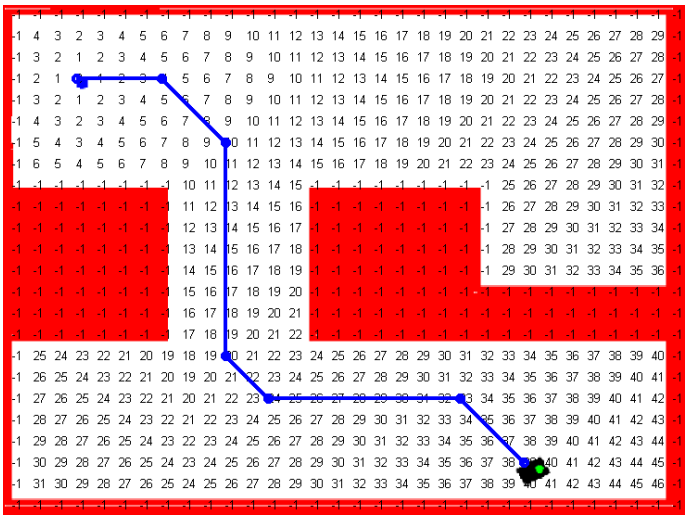
Une pondération de ($As = 0.45, Ad = 0.55$) est utilisée dans les contraintes associées au système de relations 3.4.

Pour le premier environnement, la trajectoire est constituée de 37 cellules ($nb_{cellules} = 36$) avec un Ps_{Total} de 123 ($Ps_{Total} = 123$) et présente un ratio $RS = 123/37 = 3.32$. On note la présence d'un seul conflit, dont l'apparition correspond au changement de couloir, la navigation se fera en toute sécurité sur des cellules présentant un potentiel de sécurité supérieur à 3 et qui est appelé à s'améliorer après l'opération de lissage.

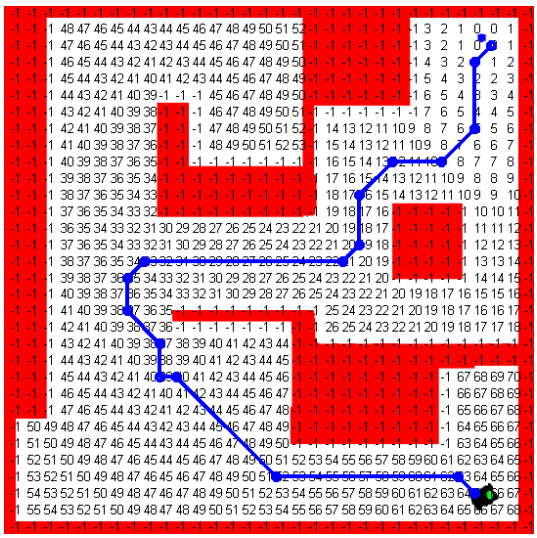
Pour le deuxième environnement, la trajectoire est constituée de 59 cellules ($nb_{cellules} = 55$) avec un Ps_{Total} de 168 ($Ps_{Total} = 168$) et présente un ratio $RS = 168/59 = 2.85$. L'irrégularité de la trajectoire obtenue traduit la gestion, presque en permanence, du conflit entre les directions proposées à la navigation.

Technique de génération de trajectoire en mode pondéré

Même en l'absence de conflits, le système de relations 3.4 est appliqué en permanence pour déterminer les directions de déplacements.



(a) Chemin pour l'environnement 1



(b) Chemin pour l'environnement 2

FIGURE 3.21 – Cartographie de l'environnement en mode dégradé

Les pondérations utilisées pour le couple(As, Ad) sont respectivement $(0.4, 0.6)$

Les résultats obtenus pour le premier environnement donnent, une trajectoire constituée de 32 cellules ($nb_{cellules} = 32$) avec un Ps_{Total} de 94 ($Ps_{Total} = 94$) et présente un ratio $RS = 94/32 = 2.94$. On note que la gestion du conflit, lié au changement de couloir, est traitée précocement. La trajectoire obtenue est plus rapide et plus régulière. La navigation se fera sur des cellules présentant un ratio de sécurité plus faible.

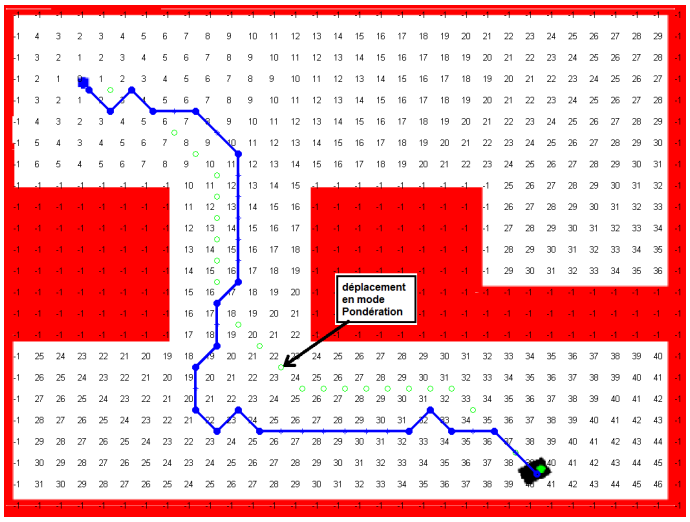
Pour le deuxième environnement, la trajectoire est constituée de 55 cellules ($nb_{cellules} =$

55) avec un Ps_{Total} de 143 ($Ps_{Total} = 143$) et présente un ratio de $RS = 143/55 = 2.6$. On note une nette amélioration de la régularité et de la rapidité de la trajectoire obtenue.

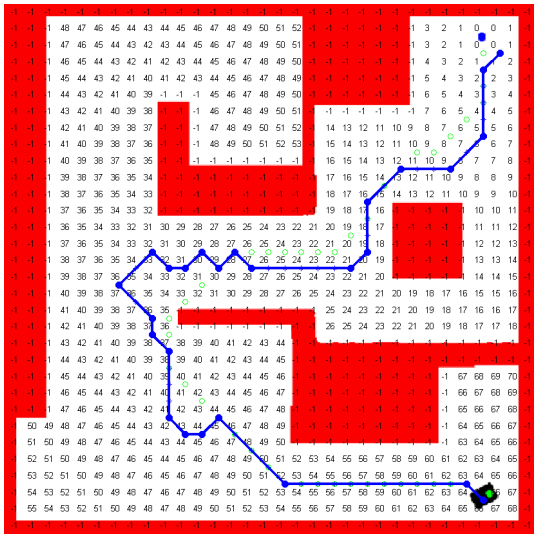
Technique de génération de trajectoire en mode combiné

L'intérêt de cette technique est de continuer de naviguer sur des lignes de haute valeur de sécurité au sein d'un couloir et n'entamer le changement de couloir, en donnant la priorité à la vitesse d'évolution vers la cellule cible, que lorsque la différence de potentiel de descente entre les cellules proposées à la navigation par les deux méthodes devient importante.

Cette trajectoire a réalisé 36 itérations pour un $Ps_{Total} = 114$ pour l'environnement 1 et 58 itérations pour un $Ps_{Total} = 157$ pour l'environnement 1.



(a) Chemin pour l'environnement 1



(b) Chemin pour l'environnement 2

FIGURE 3.22 – Cartographie de l'environnement en mode Mixe

On remarque que les trois techniques effectuent une planification de trajectoire sans aucun problème en respectant les contraintes exigées par chacune d'elle pour les deux environnements ce qui explique leur robustesse. Les trajectoires proposées par les méthodes pondérée et combinée sont plus régulières, plus rapides que celles proposées par la méthode normale, néanmoins elles présentent un ratio de sécurité plus faible.

3.4.2 Établissement de trajectoires pour des environnements réels

Présentation des environnements réels

Les sites à partir desquels ces environnements, utilisés pour valider le choix dans les techniques de génération de trajectoire, ont été extraits, sont précisés dans les légendes.

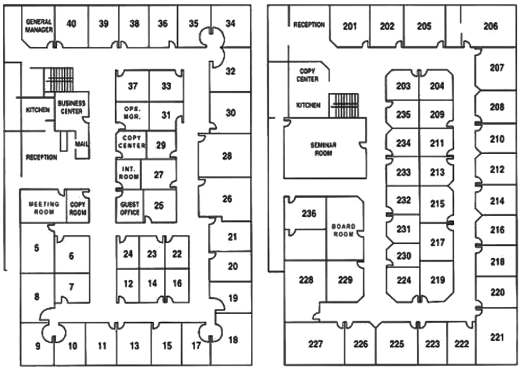
Neuf environnements correspondant à des bureaux d'étude et d'affaire, à des centres commerciaux et à des musées ont été retenus (figure 3.23 et 3.24).

Comparaison des de la qualité des trajectoires obtenues

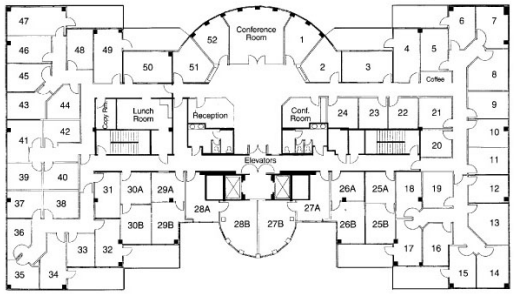
le premier tableau (tableau 3.1) consigne les résultats obtenus pour le Ps_{Total} pour chaque environnement et pour chaque méthode et le second tableau (tableau 3.2) contient les informations sur la longueur de la trajectoire (nombre de cellules), alors que le troisième tableau (tableau 3.3) donne les résultats obtenus en termes de ratio de sécurité (RS). Les informations contenues dans ces tableaux sont représentées sous formes de barres graphes dans les Figure 3.25, Figure 3.26 et Figure 3.27.

	env0D	env0G	env1	env2	env3	env4	env5	env6	env7	env8
Conflit $As = 0,45$	607	555	216	541	1437	335	420	421	968	461
Pondération $As = 0,45$	569	555	216	512	1013	324	419	409	843	407
Pondération $As = 0,2$	289	308	184	298	471	248	242	256	506	183
Vote $As = 0,45$ $As_{virtuel} = 0,2$	571	536	215	523	1140	317	393	401	917	412

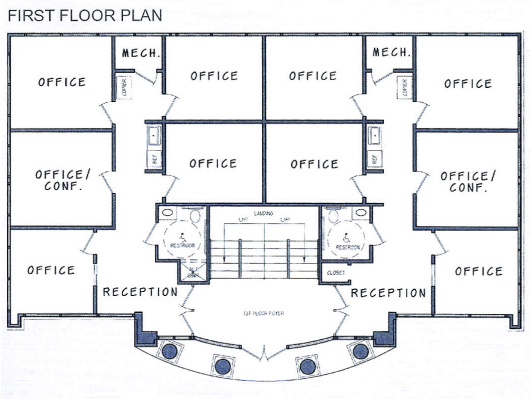
TABLE 3.1 – Résultat de la valeur de Ps_{Total} de plusieurs environnements selon chaque technique



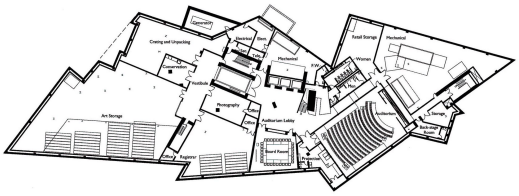
(a) Plan d'étage pour Nashville Office Space [47] (env 0D/G)



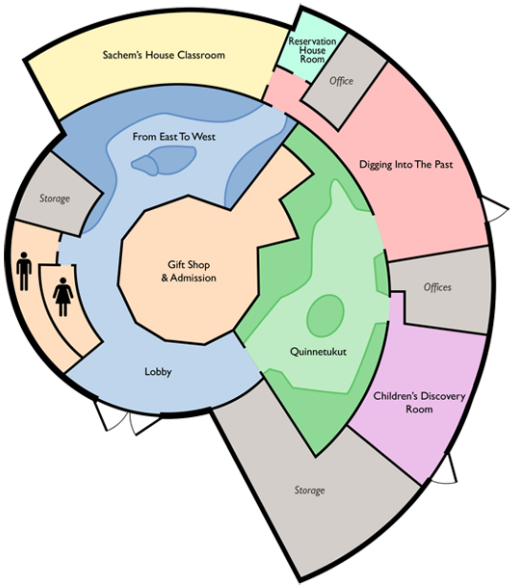
(b) Bureau commercial Plans d'étage bâtiment [46] (env 1)



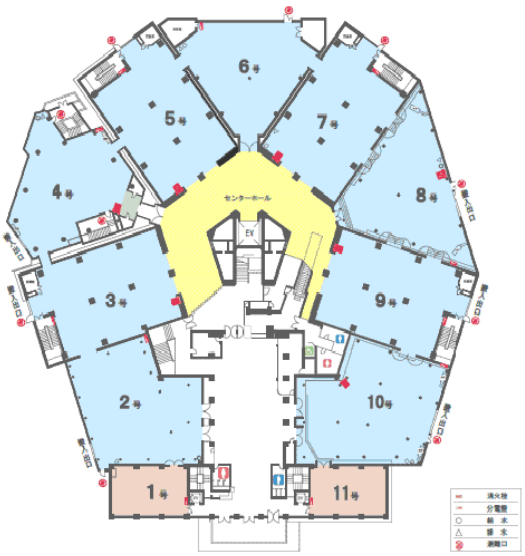
(c) Bureau commercial Plans d'étage bâtiment [15] (env 2)



(d) Musée d'art de Denver / Daniel Libeskind [39] (env 3)

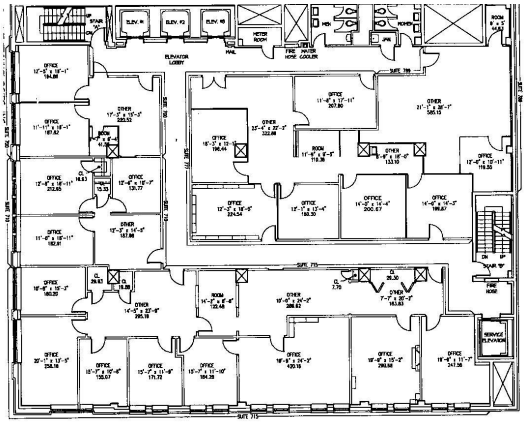


(e) L'Institut d'études amérindiennes [56] (env 4)

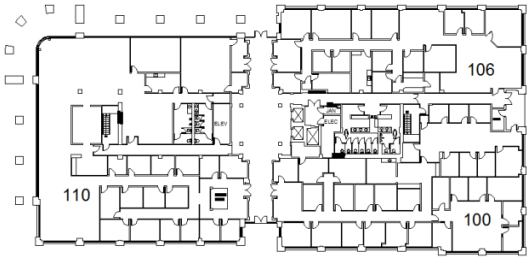


(f) Japan Foundation de la science / Musée de la Science [24] (env 5)

FIGURE 3.23 – Environnements utilisés dans les comparaisons (0 à 5)



(a) 120 Market Suites, étage 7 [57] (env 6)



(b) American Building Calculations [31] (env 7)

(c) Un bureau fonctionnel [22] (env 8)

FIGURE 3.24 – Environnements utilisés dans les comparaisons (6 à 8)

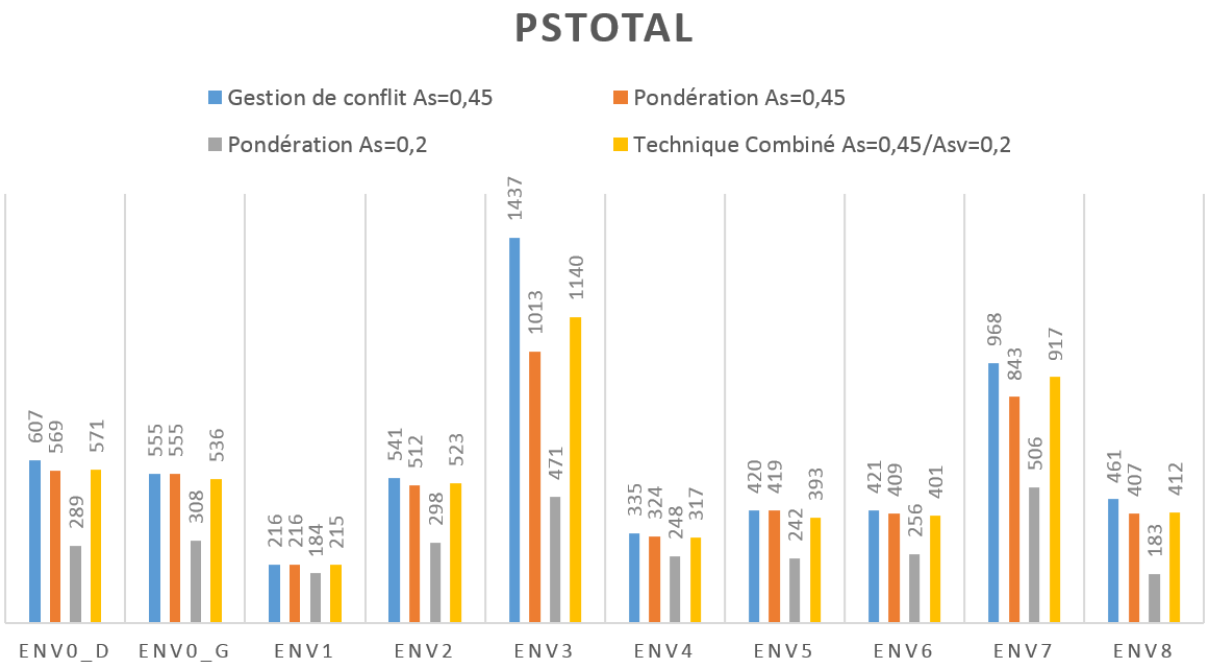


FIGURE 3.25 – Comparaison de chaque technique selon le Ps_{Total}.

	env0D	env0G	env1	env2	env3	env4	env5	env6	env7	env8
Conflit $As = 0,45$	161	155	153	133	270	79	126	141	273	103
Pondération $As = 0,45$	159	155	153	132	243	79	126	141	269	103
Pondération $As = 0,2$	146	144	153	119	216	73	109	134	254	84
Vote $As = 0,45$ $As_{virtuel} = 0,2$	160	155	153	133	253	79	123	141	274	102

TABLE 3.2 – Résultat de nombre d'itérations de plusieurs environnements selon chaque technique

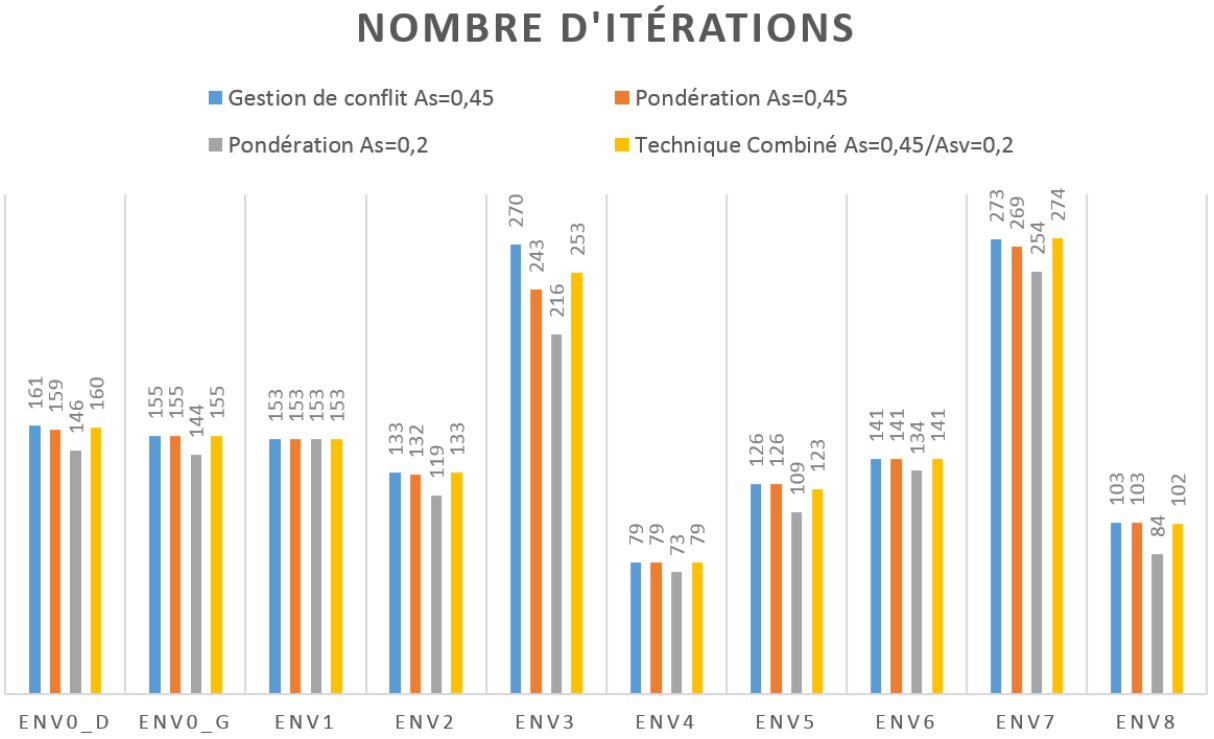


FIGURE 3.26 – Comparaison de chaque technique selon le nombre d'itérations.

	env0D	env0G	env1	env2	env3	env4	env5	env6	env7	env8
Conflit $As = 0,45$	3.77	3.58	1.41	4.07	5.32	4.24	3.33	2.99	3.55	4.48
Pondération $As = 0,45$	3.58	3.58	1.41	3.88	4.17	4.10	3.33	2.90	3.13	3.95
Pondération $As = 0,2$	1.98	2.14	1.20	2.50	2.18	3.40	2.22	1.91	1.99	2.18
Vote $As = 0,45$ $As_{virtuel} = 0,2$	3.57	3.46	1.41	3.93	4.51	4.01	3.20	2.84	3.35	4.04

TABLE 3.3 – Ratio de Sécurité de plusieurs environnements selon chaque technique

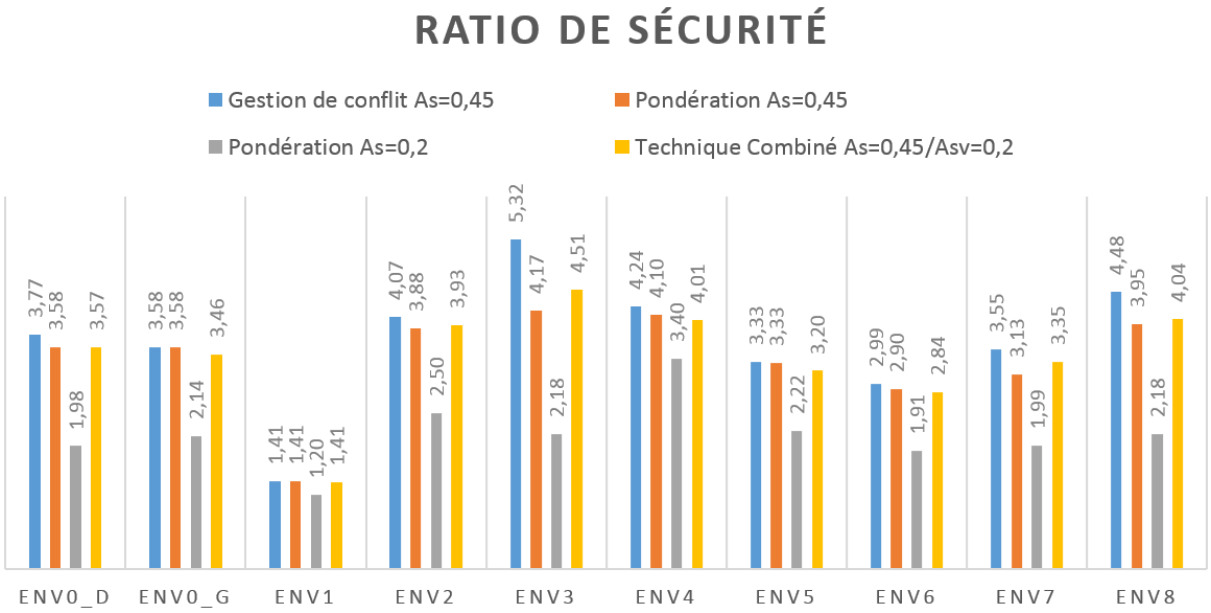
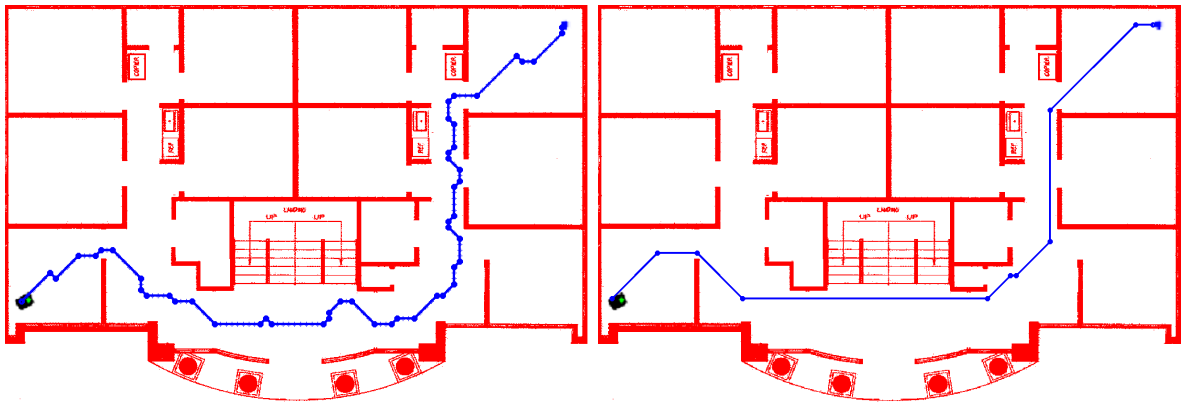
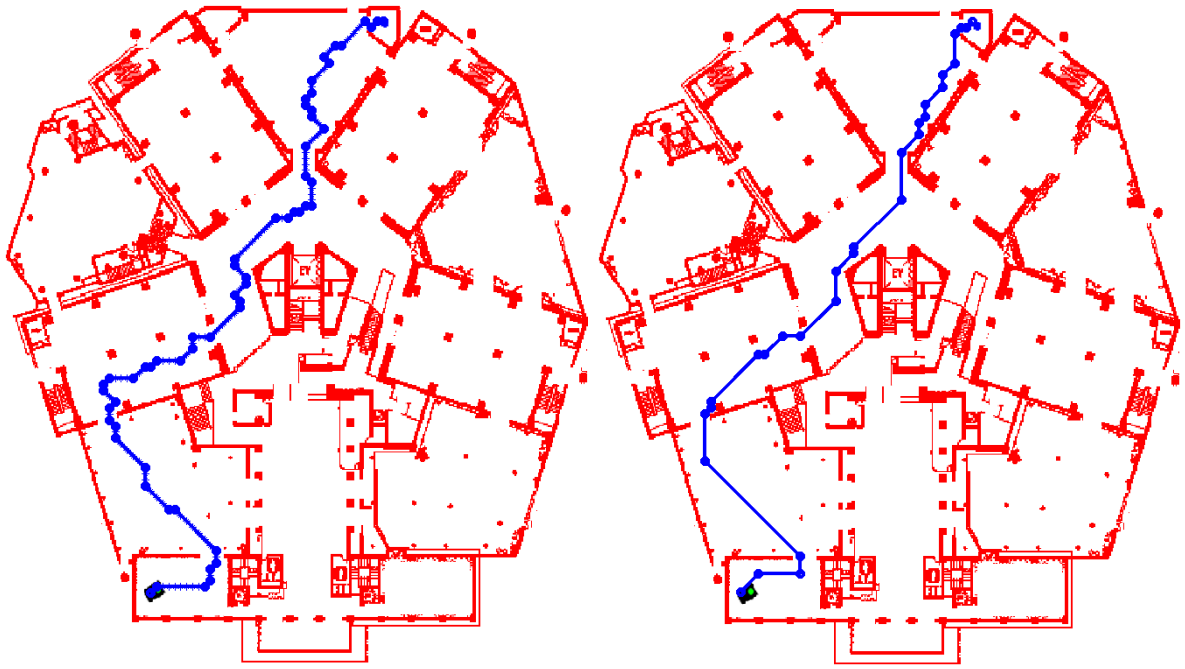


FIGURE 3.27 – Comparaison de chaque technique selon le Ratio de Sécurité.



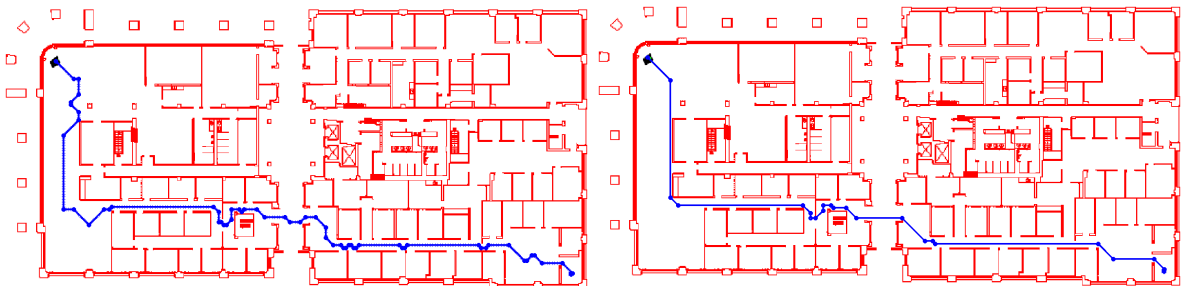
(a) Plan 2 ($As = 0.45$) (b) Plan 2 ($As = 0.2$)

FIGURE 3.28 – Résultats des techniques de navigation sur l'environnement 2 avec le facteur de sécurité $As = 0.45$ (gauche) et $As = 0.2$ (droit)



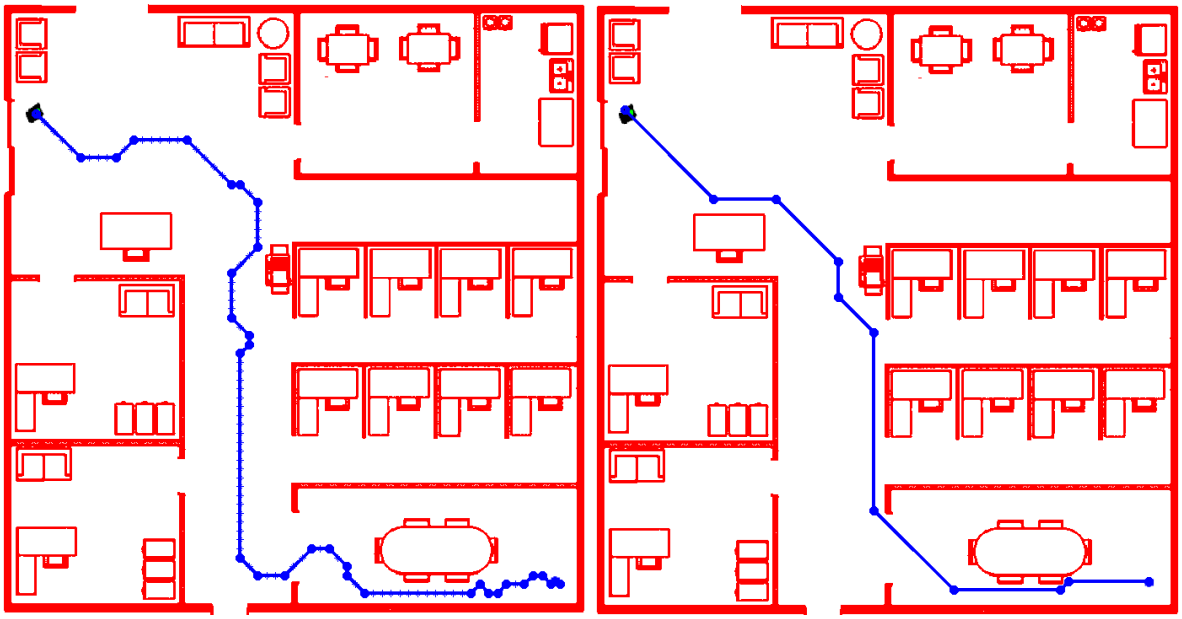
(a) Plan 5 ($As = 0.45$) (b) Plan 5 ($As = 0.2$)

FIGURE 3.29 – Résultats des techniques de navigation sur l'environnement 5 avec le facteur de sécurité $As = 0.45$ (gauche) et $As = 0.2$ (droit)



(a) Plan 7 ($As = 0.45$) (b) Plan 7 ($As = 0.2$)

FIGURE 3.30 – Résultats des techniques de navigation sur l'environnement 7 avec le facteur de sécurité $As = 0.45$ (gauche) et $As = 0.2$ (droit)



(a) Plan 8 ($As = 0.45$) (b) Plan 8 ($As = 0.2$)

FIGURE 3.31 – Résultats des techniques de navigation sur l'environnement 8 avec le facteur de sécurité $As = 0.45$ (gauche) et $As = 0.2$ (droit)

La nature de l'environnement (c'est-à-dire sa complexité : nombre de croisements, largeur des couloirs) influe énormément sur le Ps_{Total} quelle que soit la technique utilisée

La même remarque est valable aussi pour la longueur de la trajectoire, c'est-à-dire si on veut atteindre un objectif le plus rapidement possible, donc effectuer un nombre minimum de déplacements, il ne faut pas être très exigeant au niveau de la sécurité du déplacement.

Les figures 3.28, 3.29, 3.30, et 3.31 illustrent des trajectoires pour des coefficients de sécurité différents, il est apparent que le chemin devient rapide mais avec des considérations minimales pour la sécurité pour un coefficient de sécurité As minimal.

3.5 Conclusion

Ce chapitre présentait l'établissement de trajectoires alliant sécurité et rapidité sur les environnements utilisés. Leurs implémentions, d'une part sur deux sociétés d'agents mobiles logiciels : l'une développée sur MATLAB et l'autre sur la plate forme MobotSim et d'autre part sur une société d'agents réels composée de robots types POB-Bot, fait l'objet du chapitre suivant.

Chapitre 4

Simulation et Essais pratiques

4.1 Introduction

Les techniques de modélisation des environnements en zones connexes, les modes de génération de trajectoires, et les techniques de décontamination, présentées dans les deux chapitres précédents, sont implémentés dans ce chapitre, sur une société d'agents mobiles logiciels développée sur la plate-forme MobotSim [42] d'une part, et sur une société d'agents réels composée de robots types POB-Bot [48] d'autre part. L'environnement retenu, pour les essais pratiques, tient compte du nombre réduit de robots dont on dispose, mais présente suffisamment de caractéristiques intéressantes pour mettre en valeur les méthodes développées pour décontaminer un environnement d'un intrus ou localiser une personne en danger pour lui apporter assistance, en optimisant le nombre de ressources robots mobilisés pour la réussite de l'opération.

4.2 Société d'agents mobiles logiciels

4.2.1 présentation de la plate forme MobotSim

MobotSim [42] est un simulateur 2D configurable de robots mobiles à roues différentiel. Il dispose d'une interface graphique où les robots et les objets sont facilement configurés et un éditeur BASIC intégré pour le développement de la simulation.

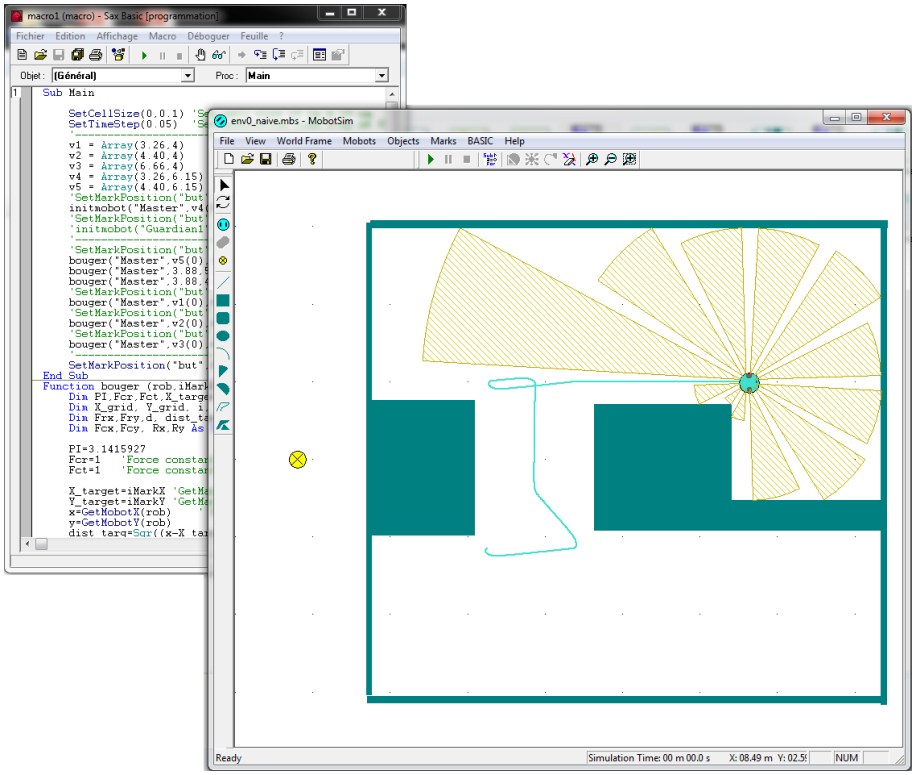


FIGURE 4.1 – Plate-forme MobotSim

L'éditeur Basic, avec lequel l'utilisateur peut écrire des macros en utilisant des fonctions spécifiques pour obtenir des informations sur les coordonnés des Mobots, sur les données de capteurs et pour régler la vitesse et la conduite des Mobots, permet l'expression de la puissance et de la facilité de langage BASIC pour programmer des techniques de navigation.

MobotSim a été développé en pensant aux chercheurs, aux étudiants, aux roboticiens et aux amateurs qui veulent concevoir, tester et simuler des robots mobiles dans les thèmes de recherche comme les techniques de navigation autonome, d'évitement d'obstacles, de l'intelligence artificielle, l'intégration des capteurs de données, etc.

4.2.2 Choix de l'environnement

Le robot utilisé à l'implémentation est un robot mobile à deux roues différentielles (figure 4.2). les environnement ont été décomposé en cellules carrés de 10 cm.

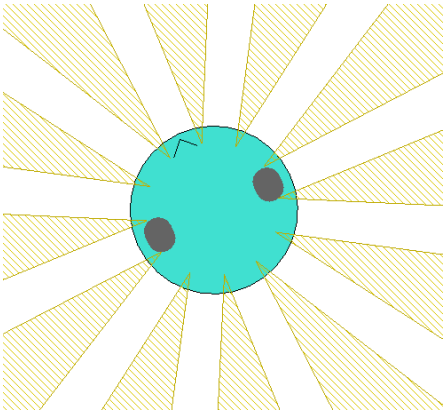
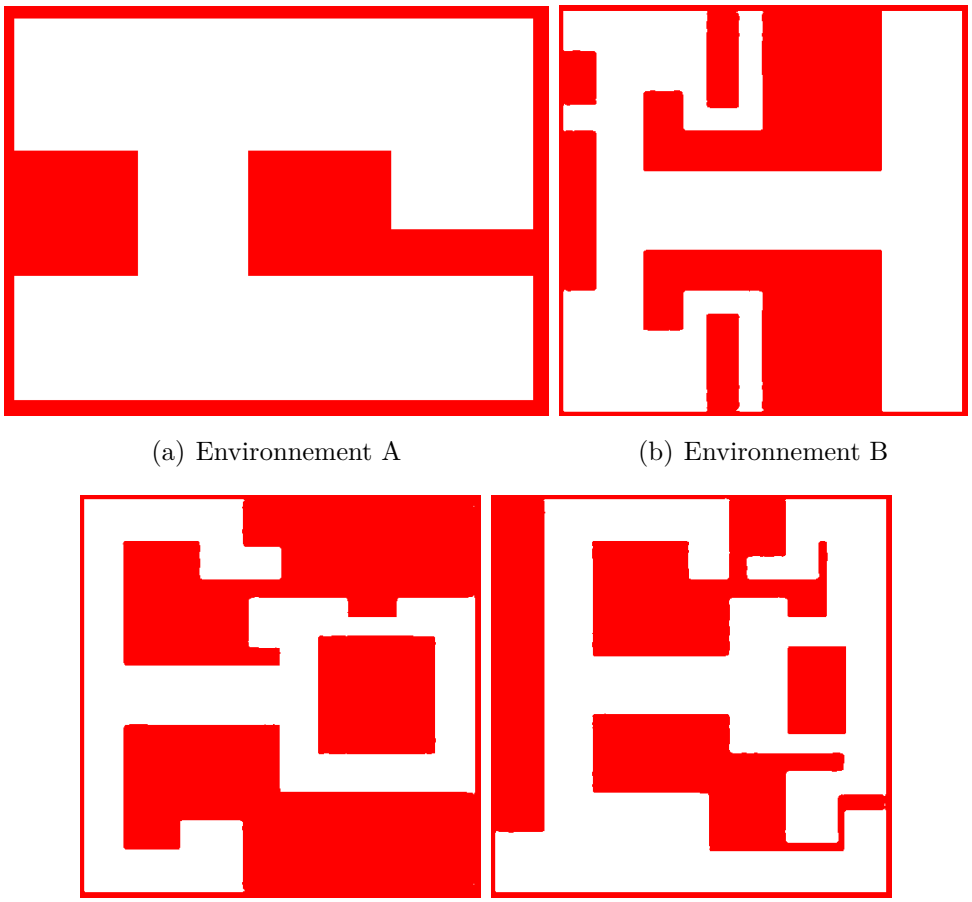


FIGURE 4.2 – Le robot Mobot

Description	**Valeur**
Diamètre de la plate-forme	25 cm
Distance entre les roues	18 cm
Diamètre des roues	5 cm

TABLE 4.1 – Caractéristiques géométriques du robot Mobot

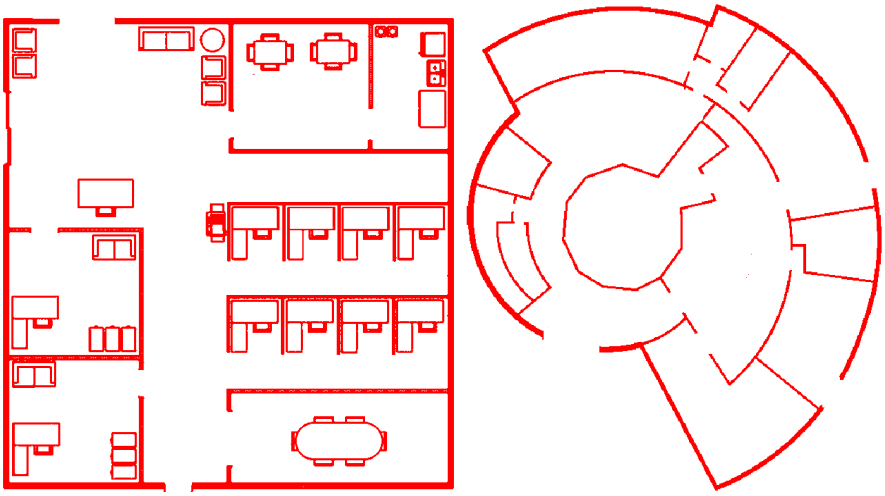
Six (06) environnements ont été retenus ; Les quatre premiers ont été conçus pour mettre en évidence la diversité des graphes et de la difficulté d'établissement des trajectoires de navigation d'une part, et la robustesse des algorithmes développés pour contourner ces difficultés d'autre part, alors que les deux derniers représentent des espaces publics tels que des centres de recherche ou des banques.



(a) Environnement A (b) Environnement B

(c) Environnement C (d) Environnement D

FIGURE 4.3 – Environnements conçus

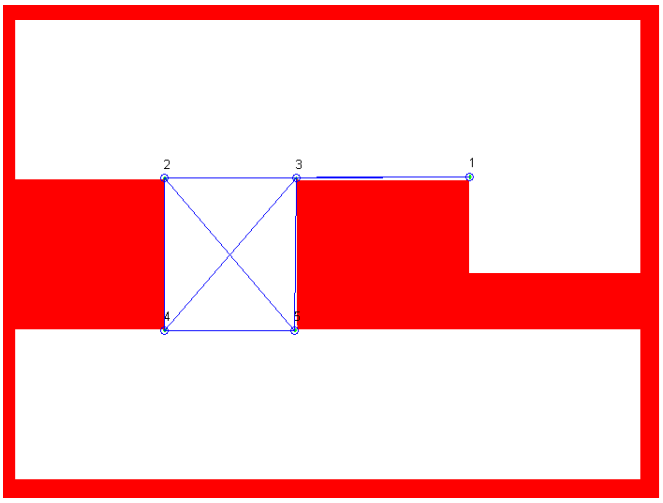


(a) Environnement E (b) Environnement F

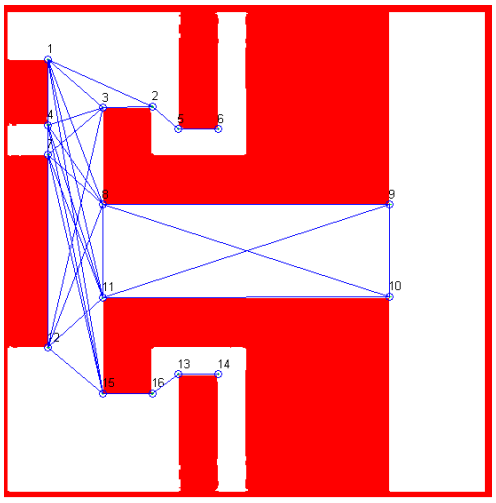
FIGURE 4.4 – Environnements réels

4.2.3 Modélisation de l'environnement en zones connexes

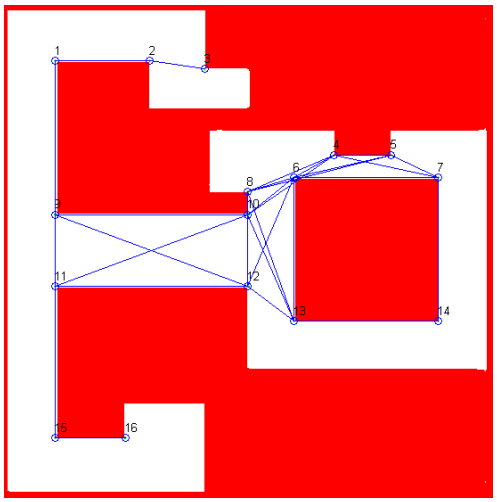
Les graphes de visibilité pour les six environnements sont représentés dans les figures 4.5 et 4.5 :



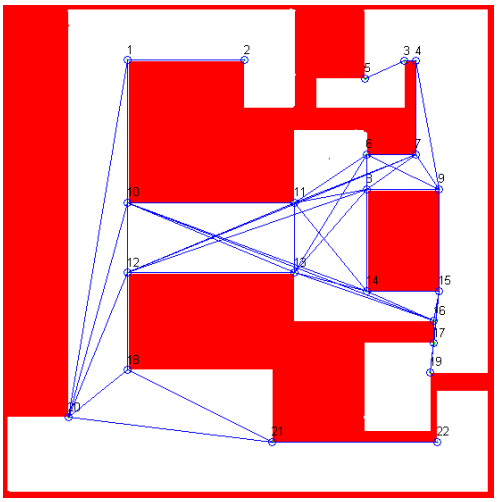
(a) Environnement A



(b) Environnement B

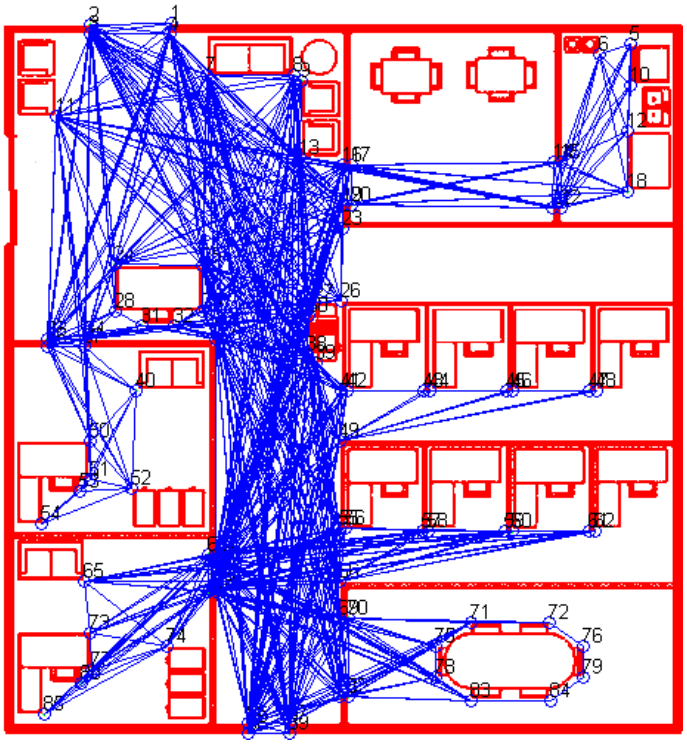


(c) Environnement C

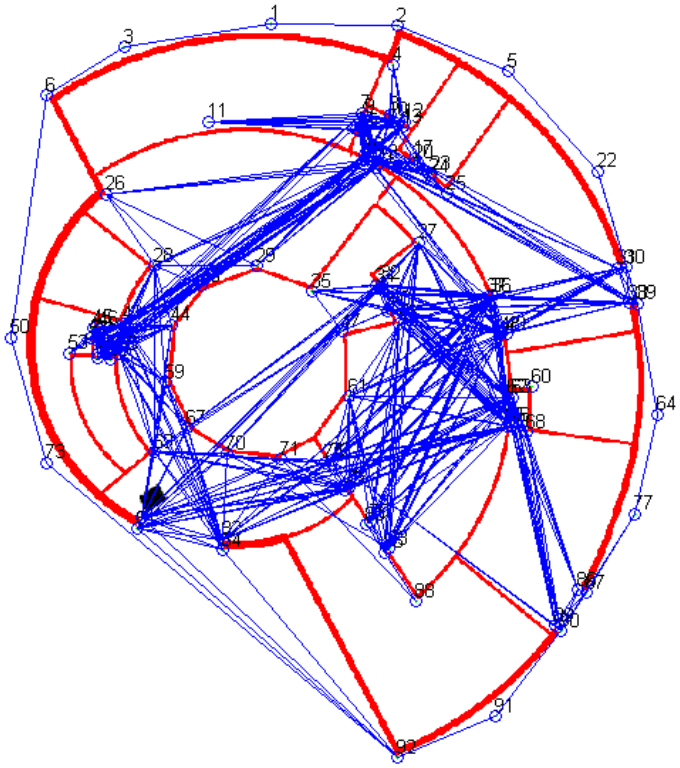


(d) Environnement D

FIGURE 4.5 – Graphes de visibilité pour les environnements conçus.



(a) Environnement E

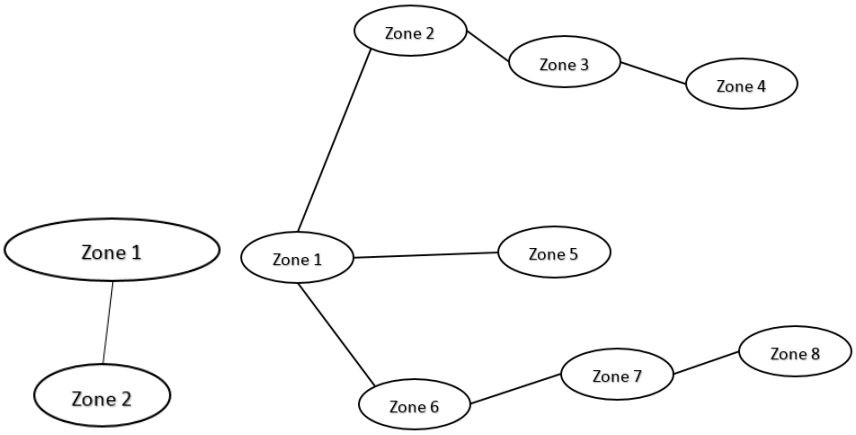


(b) Environnement F

FIGURE 4.6 – Graphes de visibilité pour les environnements réels

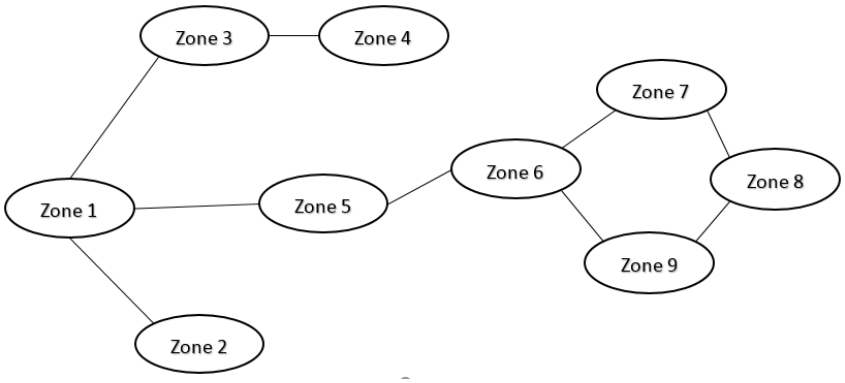
A partir des graphes de visibilité, les graphes de zones sont établis et sont présentés dans les figures 4.8 et 4.8. Le graphe de l'environnement (A) est très simple d'un point de

vue décontamination car son graphe est composé de deux zones reliées, son intérêt réside dans la mise en évidence des techniques de navigation et de leur robustesse à trouver des trajectoires sûres et rapides. Le graphe de l'environnement (B) comporte une zone critique, et les deux techniques de décontamination se valent du point de vue utilisation des ressources robots. L'environnement (C) comporte deux zones critiques et un cycle. Le graphe de l'environnement (D) comporte trois zones critiques, un cycle (zones : 6, 7, 8 et 9) et un faux cycle (zones : 8, 9 et 11) car les trois zones se partagent un ou plusieurs vertex. Les graphes des environnements réels (E et F) comportent plusieurs zones critiques et des cycles.

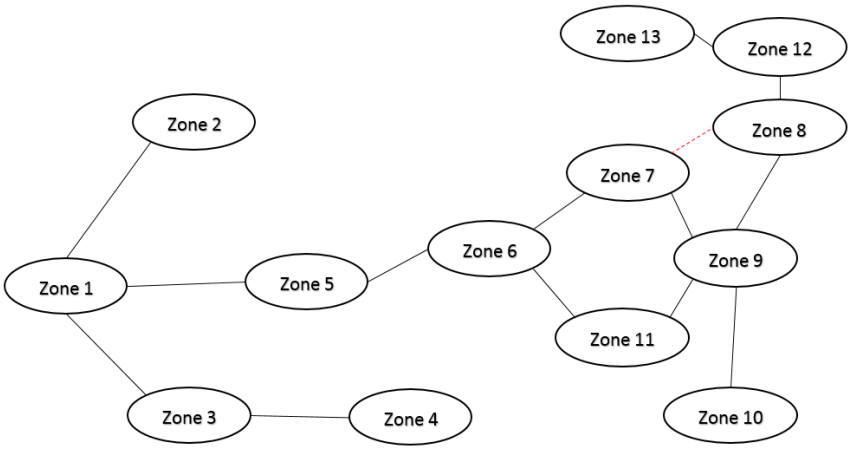


(a) Environnement A

(b) Environnement B

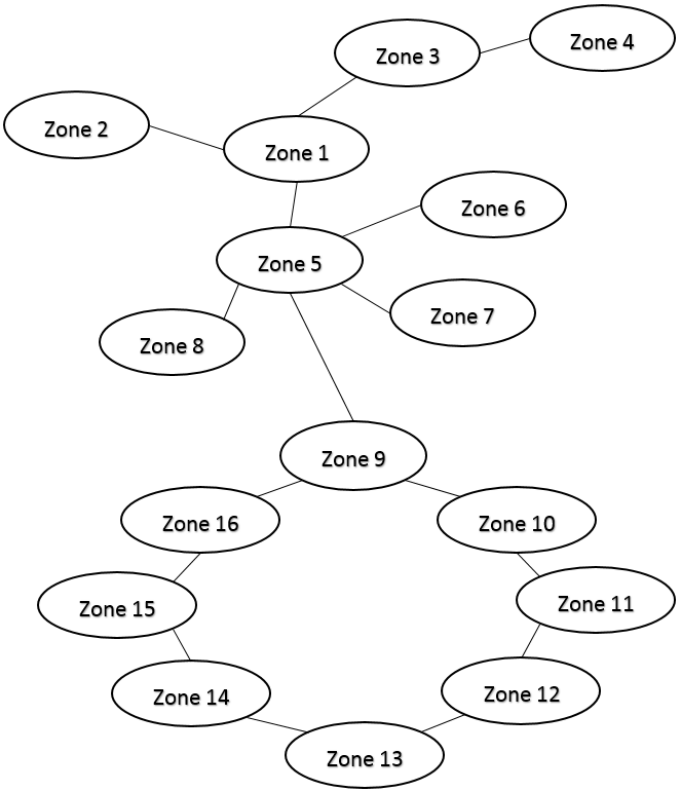


(c) Environnement C

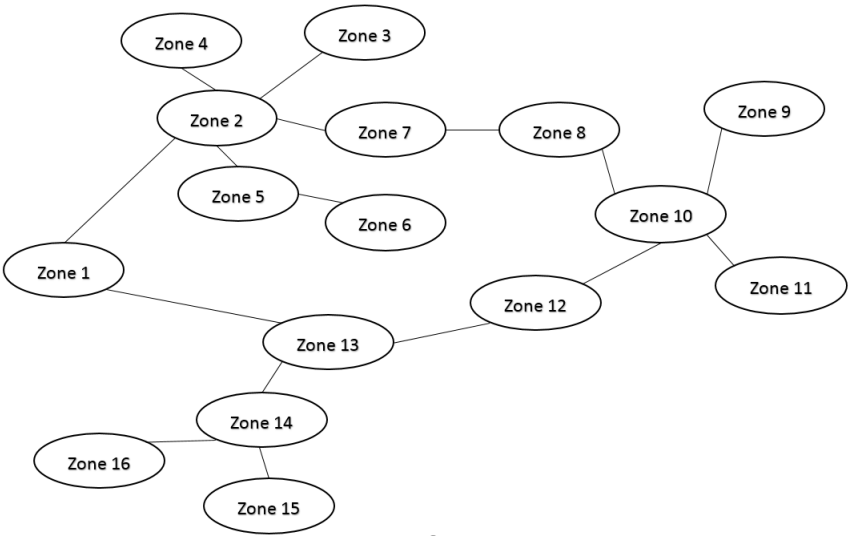


(d) Environnement D

FIGURE 4.7 – Graphes de zones pour les environnements conçus



(a) Environnement E



(b) Environnement F

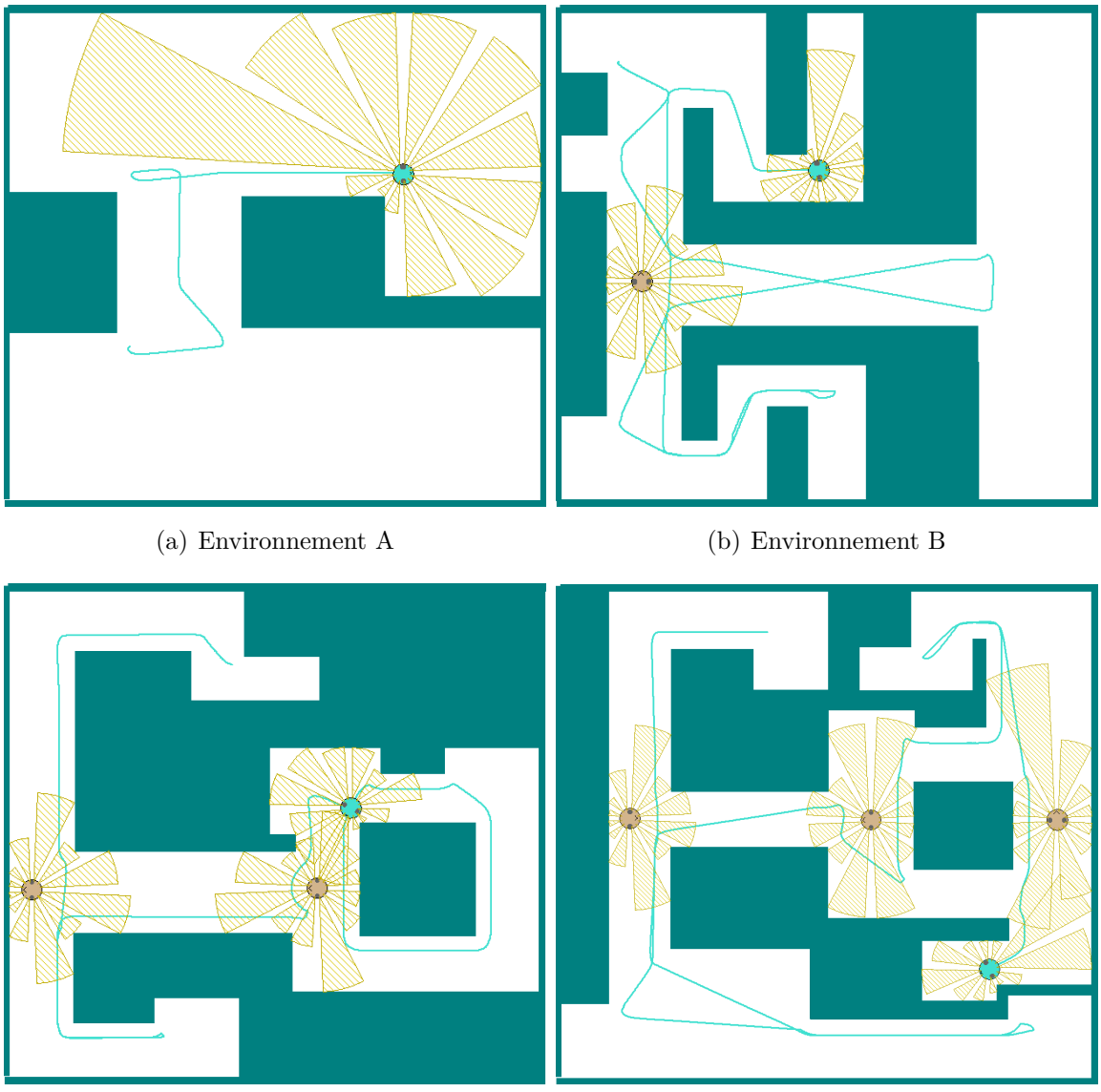
FIGURE 4.8 – Graphes de zones pour les environnements réels

4.2.4 Techniques de décontamination

Méthode Naïve

Dans la méthode naïve [36], l'agent principal parcourt l'environnement en visitant toutes les zones et place un agent de garde dans celles qui sont jugées critiques. Pour l'en-

vironnement (A) l'agent principal, seul, arrive à le décontaminer ; Pour l'environnement (B) deux agents, l'agent principal et un agent de garde, sont nécessaires à sa décontamination ; Pour l'environnement (C), trois agents sont réquisitionnés pour sa décontamination ; La décontamination des environnements (D et E) nécessite quatre agents et l'environnement (F) exige cinq agents. Sur les figures 4.9 et 4.10, l'agent principal est représenté en vert et les agents de garde en couleur marron-beige.



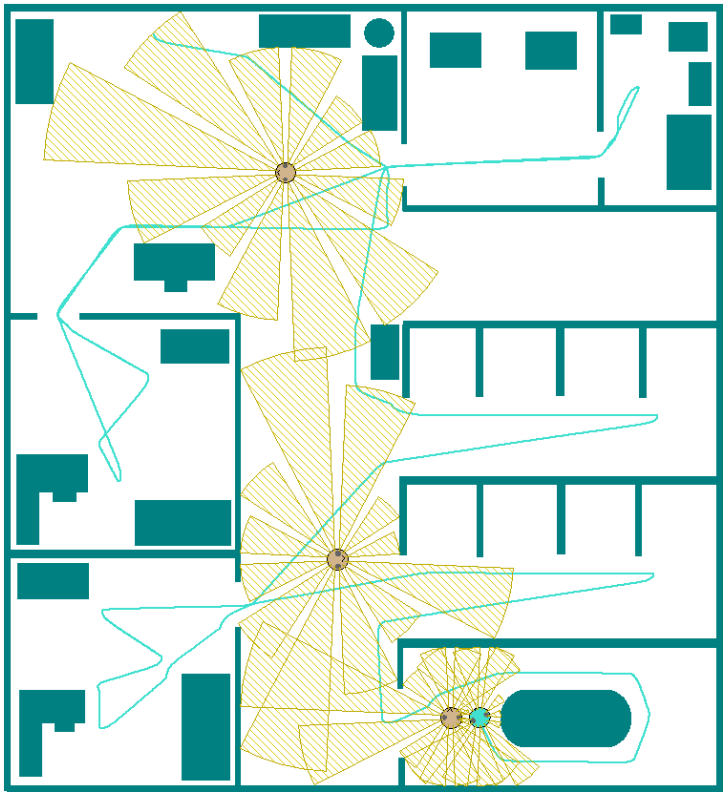
(a) Environnement A

(b) Environnement B

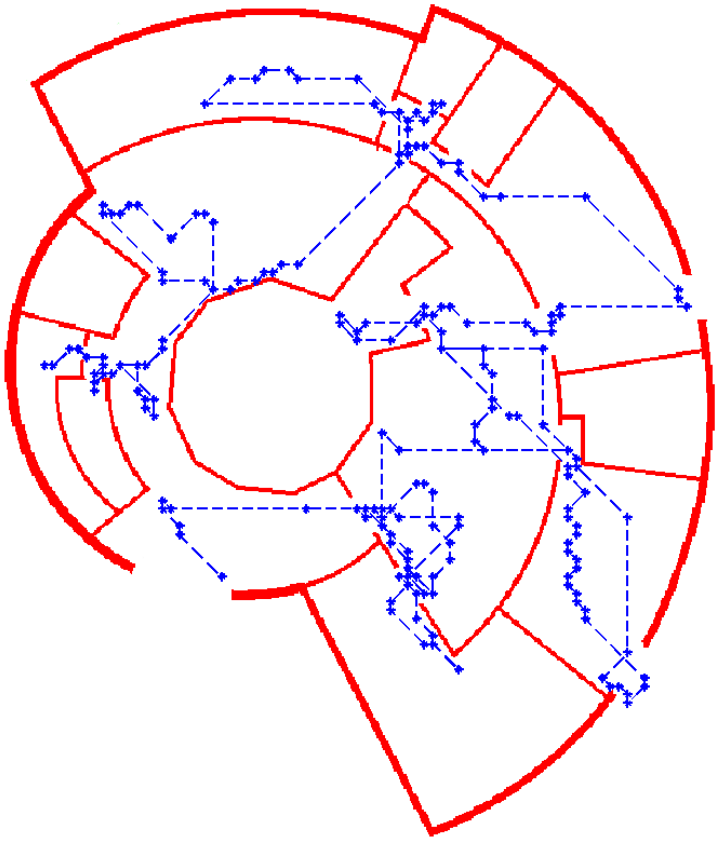
(c) Environnement C

(d) Environnement D

FIGURE 4.9 – Application de la méthode naïve sur les environnements conçus



(a) Environnement E



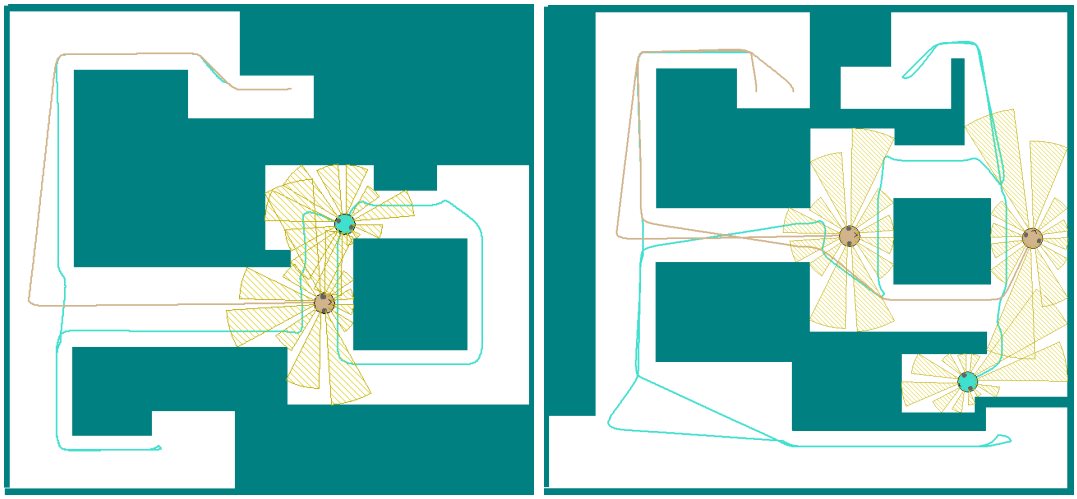
(b) Environnement F

FIGURE 4.10 – Application de la méthode naïve sur les environnements réels

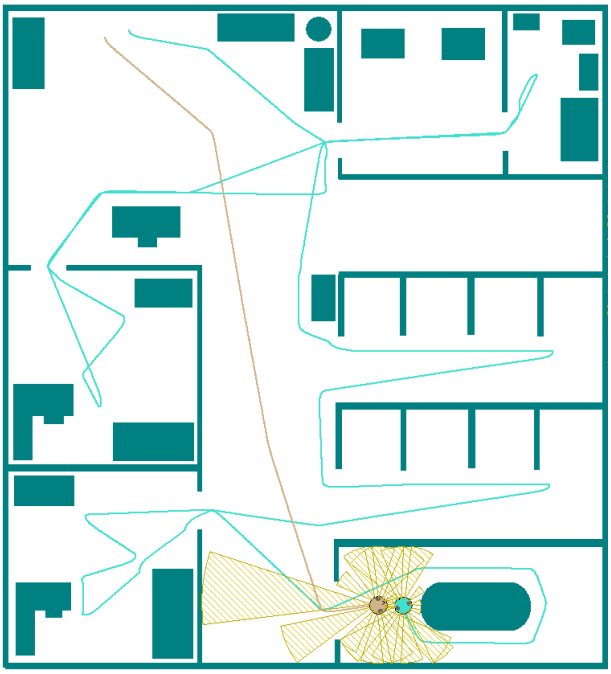
La représentation de l'environnement (F) est relativement difficile sur la plate-forme MobotSim, le parcours de l'environnement par l'agent principal, simulé sur MATLAB, est représenté en couleur bleue

Technique améliorée

La technique améliorée minimise le nombre d'agents nécessaires pour l'opération de décontamination ; Le parcours de l'environnement, par l'agent principal, commence toujours à partir d'une zone feuille. Dans le tableau 4.2, sont consignés les résultats de la simulation, montrant le nombre d'agents nécessaires à la décontamination des différents environnements, par les deux méthodes naïve et améliorée. Seuls les environnements(C, D et E), où le gain en agents de garde est conséquent, sont représentés dans la figure 4.11.



(a) Environnement A (b) Environnement B



(c) Environnement C

FIGURE 4.11 – Application de la méthode améliorée sur les les environnements (c, d, et e)

On note que l'intérêt de la méthode améliorée devient évident dans la décontamination des environnements ayant un nombre de zones critiques égal ou supérieur à deux. L'optimisation des ressources robots pour les environnements réels (E et F) est essentielle et la méthode améliorée constitue une solution adaptée, comme le montre le tableau 4.2 pour le cas de l'environnement (E et F), où le gain en ressource robots est, pour certains environnements, très important.

	Méthode naïve	**Méthode améliorée**	**Gain en agents mobiles**
Environnement a	1	1	0
Environnement b	2	2	0
Environnement c	3	2	1
Environnement d	4	3	1
Environnement e	4	2	2
Environnement f	5	3	2

TABLE 4.2 – Comparaison des nombres d'agents nécessaires

4.2.5 Réseaux de Petri de validation

L'environnement réel (e), où le gain en ressource robots est important, est retenu pour l'étude de la validation du nombre d'agents robots nécessaire à la décontamination par l'outil des réseaux de Petri. Le réseau de Petri modélisant le graphe de zones est donné dans la figure 4.12. Et pour chaque place de ce réseau on associe trois sous-réseaux de Petri modélisant respectivement les comportements *guarded*?, *contam*? et *decontamAll*?. Pour des raisons de commodité, le réseau de Petri global est présenté en annexe (figure A.2, page 117), il comporte 112 places et 144 transitions (conforme aux résultats fournis par l'équation 2.17). La validation du nombre d'agents se fait en suivant simultanément le marquage du réseau de Petri associé au graphe de zones qui est un marquage p-invariant (le nombre de jetons est égal au nombre d'agents robots) et le marquage associé aux places $contamA_i$? du sous-réseau *contam*? dont le marquage initial est un vecteur unité traduisant la contamination de toutes les zones et le marquage final est le vecteur nul traduisant la décontamination totale de l'environnement.

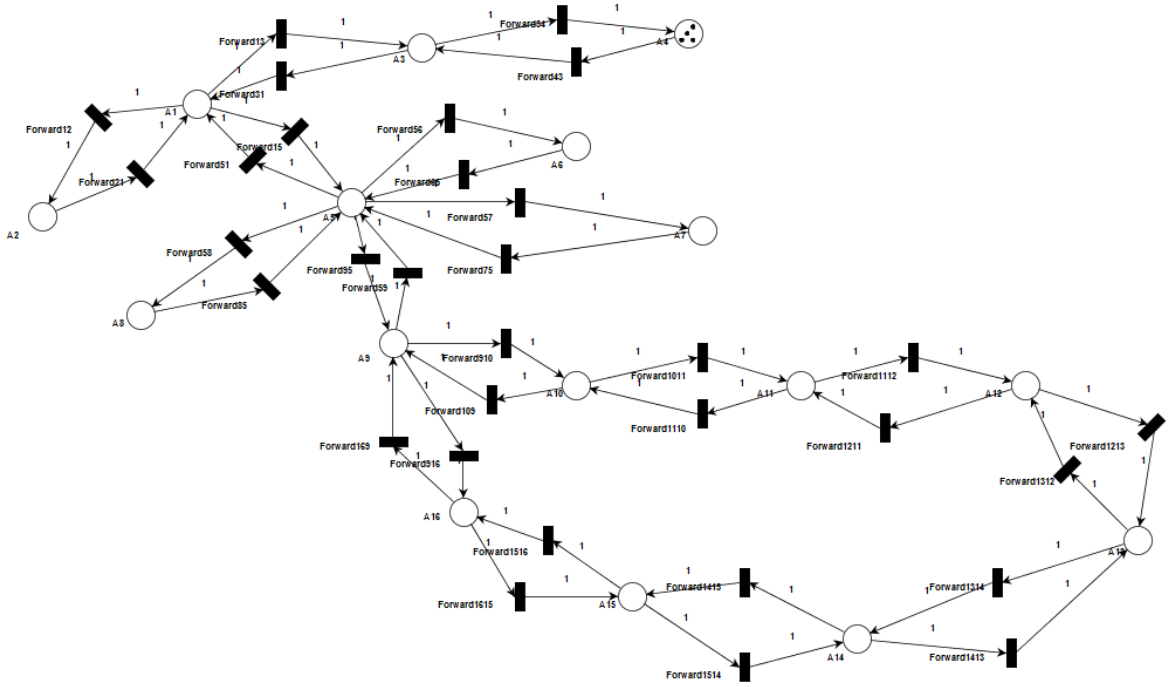


FIGURE 4.12 – Réseau de Petri de zones de l'environnement e.

Pour les deux méthodes de décontamination, les marquage des places A_i et $contamA_i?$ sont représentés dans une matrice à deux colonnes ; Le marquage initial des places A_i est donné par le vecteur colonne de gauche, dont tous les éléments sont nuls, sauf un et sa valeur indique le nombre d'agents robots nécessaires à la décontamination. Le marquage initial des places $contamA_i?$ est placé dans le vecteur colonne de droite et il est égal au vecteur unité. Le marquage final est atteint, et le nombre d'agents robots mobilisés pour l'opération de décontamination est validé, lorsque le vecteur colonne relatif aux places $contamA_i?$ devient égal au vecteur nul.

Ci-après présente l'évolution du marquage dans la méthode naïve : du marquage initial au marquage final.

$$M_0 = \begin{bmatrix} A_1 & contamA_1 \\ A_2 & contamA_2 \\ A_3 & contamA_3 \\ A_4 & contamA_4 \\ A_5 & contamA_5 \\ A_6 & contamA_6 \\ A_7 & contamA_7 \\ A_8 & contamA_8 \\ A_9 & contamA_9 \\ A_{10} & contamA_{10} \\ A_{11} & contamA_{11} \\ A_{12} & contamA_{12} \\ A_{13} & contamA_{13} \\ A_{14} & contamA_{14} \\ A_{15} & contamA_{15} \\ A_{16} & contamA_{16} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 4 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \overset{\sigma_1}{\rightarrow} \begin{bmatrix} 4 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \overset{\sigma_2}{\rightarrow} \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 3 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \overset{\sigma_3}{\rightarrow} \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 2 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \overset{\sigma_4}{\rightarrow} \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} = M_{final} \tag{4.1}$$

Avec :

— $\sigma_1 = 4 \times Forward_{43} \rightarrow 4 \times Forward_{31}$
— $\sigma_2 = 1 \times Forward_{12} \rightarrow 1 \times Forward_{21} \rightarrow 3 \times Forward_{15}$
— $\sigma_3 = 1 \times Forward_{56/65} \rightarrow 1 \times Forward_{57/75} \rightarrow 1 \times Forward_{58/85} \rightarrow 2 \times Forward_{59}$
— $\sigma_4 = 1 \times Forward_{910} \cdots 1 \times Forward_{1516}$

Ci-après présente l'évolution du marquage dans la méthode améliorée : du marquage initial au marquage final.

$$M_0 = \begin{bmatrix} A_1 & contamA_1 \\ A_2 & contamA_2 \\ A_3 & contamA_3 \\ A_4 & contamA_4 \\ A_5 & contamA_5 \\ A_6 & contamA_6 \\ A_7 & contamA_7 \\ A_8 & contamA_8 \\ A_9 & contamA_9 \\ A_{10} & contamA_{10} \\ A_{11} & contamA_{11} \\ A_{12} & contamA_{12} \\ A_{13} & contamA_{13} \\ A_{14} & contamA_{14} \\ A_{15} & contamA_{15} \\ A_{16} & contamA_{16} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 2 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \xrightarrow{\sigma_1} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 2 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \xrightarrow{\sigma_2} \begin{bmatrix} 2 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \xrightarrow{\sigma_3} \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \xrightarrow{\sigma_4} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 2 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

$$\xrightarrow{\sigma_5} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \xrightarrow{\sigma_6} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 2 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \xrightarrow{\sigma_7} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} = M_{final} \quad (4.2)$$

Avec :

— $\sigma_1 = 1 \times Forward_{43} \rightarrow 1 \times Forward_{31}$
— $\sigma_2 = 1 \times Forward_{43} \rightarrow 1 \times Forward_{31}$
— $\sigma_3 = 1 \times Forward_{12/21} \rightarrow 1 \times Forward_{15}$
— $\sigma_4 = 1 \times Forward_{15}$
— $\sigma_5 = 1 \times Forward_{56/65} \rightarrow 1 \times Forward_{57/75} \rightarrow 1 \times Forward_{58/85} \rightarrow 1 \times Forward_{59}$
— $\sigma_6 = 1 \times Forward_{59}$
— $\sigma_7 = 1 \times Forward_{910} \cdots 1 \times Forward_{1516}$

4.3 Société d'agents robots type : POB-Bot

4.3.1 Présentation du robot POB-Bot

Historique

Le robot POB-Bot est un petit robot pédagogique et ludique, réalisé par POB-Technology [48].

Bot, célèbre abréviation anglophone du mot robot, est associée ici au nom de l'entreprise POB-Technology (POB = Pieces Of Bot)qui le produit et le développe. Il a été conçu et développé au fil des années pour permettre aux étudiants et aux passionnés de la robotique de découvrir et exploiter cet univers, et de pouvoir, à partir du POB-Bot, de réaliser les projets de leur choix.

Le robot est doté de deux moteurs lui permettant de se mouvoir, d'une structure mécanique qui lui sert de support, de dispositifs de perception et bien évidement d'une carte de contrôle des servomoteurs, et gère les entrées/sorties (analogiques et numériques) des capteurs et actionneurs.

C'est pourquoi toutes les technologies constituant le POB-Bot sont des technologies standard, et, en conséquence, modifiables à souhait, ayant pour objectifs le développement d'applications basées sur :

— L'intelligence
— La vision
— La mobilité
— La perception de l'environnement
— L'action sur l'environnement
— La communication

Structure matérielle du robot

Comme le montre la figure 4.13, le robot est composé de quatre parties dont chacune a un rôle spécifique.

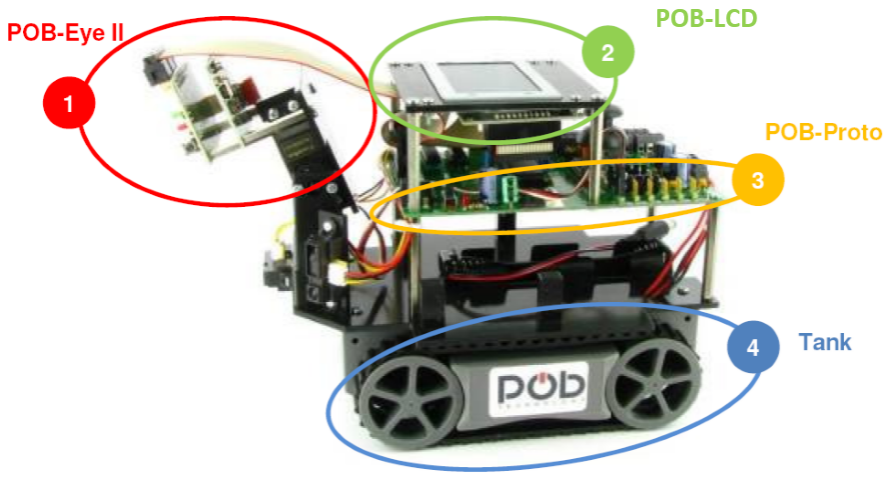


FIGURE 4.13 – Robot POB-Bot de type Golden Pack

Description	Valeur
Longueur de la plate-forme	20 cm
Largeur de la plate-forme	13 cm
Distance entre les roues	10 cm
Longueur des roues	16 cm

TABLE 4.3 – Caractéristiques géométriques du robot Mobot

Partie 1 : POB-Eye II

Le POB-Eye II est la tête du robot. Il s'agit d'un module électronique comportant non seulement la caméra, mais aussi l'essentiel des composants qui exécutent les programmes et analysent l'environnement. Véritable cerveau du POB-Bot, le POB-Eye (Figure 4.14) se charge d'effectuer tous les calculs : il exécute le programme. (Dans notre projet POB-Eye II n'est pas exploitée) C'est également lui qui réalise l'acquisition d'image via sa caméra (1). Les deux LEDs (2) situées sur la gauche de la carte sont des indicateurs : la LED rouge indique que le robot est allumé et sous tension, la LED verte est programmable, et peut donc vous servir de signal visuel. Un petit bouton de programmation utilisé pour configurer le POB-Eye se situe sur son côté droit (3).

1. Caméra
2. LEDs
3. Bouton de programmation

Description	Valeur
Processeur	ARM 7TDMI @ 60Mhz
Mémoire Flash	128 Ko
RAM	64 Ko
Entrées/Sorties	15
Bus	I2C et UART
Taille de l'image capturée	88 × 120 pixels (32Ko)
Image couleur	(1 octet par composante)
Alimentation	6v à 9v, 2A maximum
Tension des signaux logiques	3.3v (5v tolérant)
Courant des signaux logiques	10mA maximum
Dimensions	49.5 × 64.3 mm

TABLE 4.4 – Caractéristiques du processeur du POB-Eye

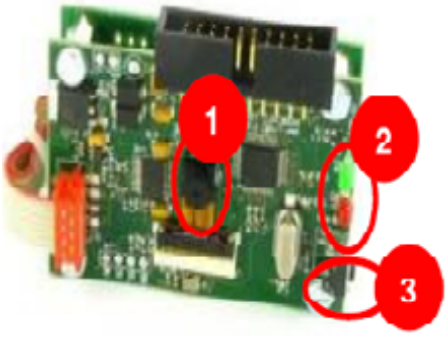


FIGURE 4.14 – POB-Eye II

Partie 2 : POB-Proto

La carte POB-Proto est l'élément moteur du robot (Figure 4.15). Cette carte contrôle les servomoteurs, et gère les entrées/sorties (analogiques et numériques) des capteurs et actionneurs. Contient 6 sorties servomoteurs et numériques (4), 4 entrées numériques (2), et 3 entrées analogiques (1). De quoi y connecter le matériel de choix. Le Joystick analogique (3) permettra de naviguer dans certains programmes mais pourra aussi être utilisé dans des programmes en tant que capteur analogique. La POB-Proto est aussi dotée du connecteur d'alimentation (6), qui permet au robot d'être raccordé à une batterie ou une prise secteur via un câble d'alimentation approprier. Enfin, on note la présence du bouton ON/OFF à coté de ce connecteur (5).

Description	Valeur
Processeur	PIC 16F877 reprogrammable
Joystick	1
Servotomoteurs	6
Moteurs	2 moteurs à courant continu par Pont en H
Entrées Analogiques	6 entrées analogiques (peuvent être configurées comme des I/O digitales)
E/S Numériques	8 I/O digitales
	6 autres I/O digitales (qui peuvent aussi contrôler 6 servomoteurs)
Pin	Pin pour alimentation (5v et +alim)
Bus	Bus I2C (avec Pob-Eye)

TABLE 4.5 – Caractéristiques du processeur du POB-Proto

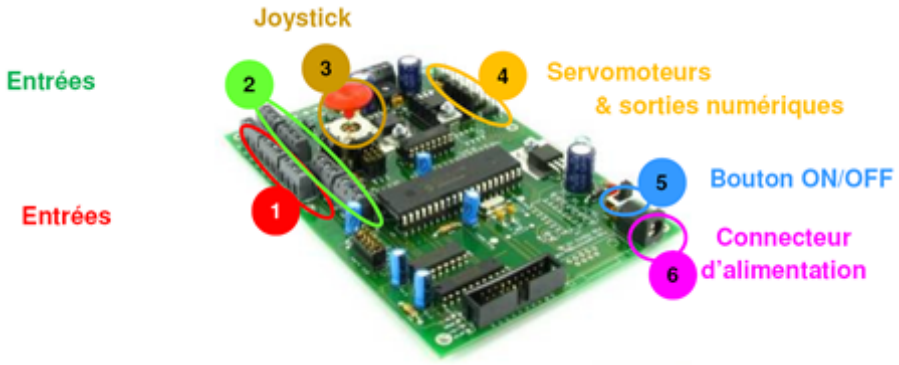


FIGURE 4.15 – Carte de commande du robot : POB-Proto

Partie 3 : POB-LCD

Le POB-LCD est un écran en noir et blanc qui permet de suivre l'évolution du programme. Loin de représenter la qualité réelle du capteur CMOS (caméra), cet écran permet toutefois d'afficher les valeurs retournées par les capteurs, les messages textes, ainsi que l'image simplifiée de la caméra, par contraste noir et blanc.

Partie 4 : tank

Notre robot est un robot mobile à chenille, le tank est la base mobile du POB-Bot. Contrôlés par la POB-Proto, servant de support du système de commande et de tous les autres éléments du robot, les deux moteurs présents dans ce tank permettent de faire avancer ou reculer les chenilles, ce qui permet au robot d'avancer, de reculer et de tourner (Figure 4.16).

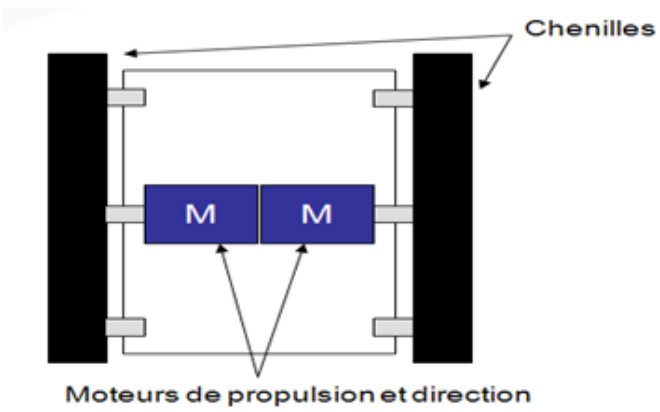
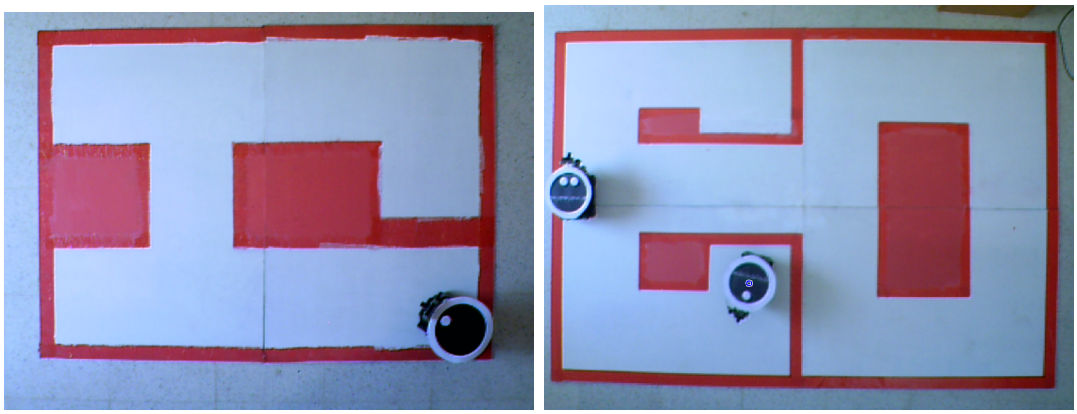


FIGURE 4.16 – Plateforme mobile du robot

4.3.2 Environnements tests

Les environnements tests retenus sont conçus en tenant compte du nombre réduit de robots dont on dispose (2 robots POB-BOT). Ils sont représentés dans les figures 4.17 ; Le premier est très simple d'un point de vue décontamination car il est constitué de deux zones, mais du fait de la présence de deux couloirs larges reliés par un couloir étroit, permet de mettre en évidence la gestion des conflits lors de la génération de trajectoire entre les directions à forte valeur de potentiel de sécurité et celles indiquant la descente rapide vers la cellule cible. Le deuxième environnement test, dans lequel existe une zone critique, permet de tester et de valider les techniques de décontamination et d'optimisation du nombre d'agents nécessaires à la réussite de cette opération, met en valeur, pour notre cas, la collaboration de deux robots pour réussir cette mission.



(a) Environnement 1 (b) Environnement 2

FIGURE 4.17 – Présentation de l'environnement réel

Dans les deux environnements utilisés, l'espace libre disponible à la navigation est en blanc et l'espace indisponible à la navigation est en rouge : obstacles (murs) (Figure 4.17).

4.3.3 Acquisition d'images vidéo par camera web

L'acquisition d'images vidéo par caméra web permet, dans un contexte offline de numériser l'environnement, car dans notre application, les agents robots évoluent dans des environnements connus, et dans un contexte online, de fournir la position courante et l'orientation de chaque agent de la société.

L'acquisition d'images est réalisée par une caméra de type Webcam, de basse résolution à partir d'une vidéo VGA dont la résolution est de 1024×768 pixels. La position courante d'un robot et son orientation dans son environnement sont réalisées, grâce à l'utilisation d'un artifice de grande simplicité, en plaçant sur le robot un grand cercle de couleur foncée (détermination de la position du robot) à l'intérieur duquel se trouvent un ou plusieurs cercles blancs placés en avant (détermination de son orientation). Cette simple technique donne d'excellents résultats en termes de temps de calcul et de robustesse vis-à-vis des perturbations dus aux changements d'éclairage. La figure 4.18 montre un robot portant, dans sa partie supérieure, le système de localisation et d'identification

(a) Robot mobile 1 (b) Robot mobile 2

FIGURE 4.18 – Robots mobiles réels

L'utilisation des toolbox *Image Acquisition* et *Image Processing* de Matlab permettent de réaliser l'acquisition et le traitement d'images, notamment les fonctions *imread* et *imfindcircles* qui permettent, respectivement, de lire une image et de manipuler son contenu dans une variable de type matrice, et de fournir les paramètres (coordonnées et rayon) de chaque cercle présent dans l'image traitée en fonction de la polarité (sombre ou claire) et d'un intervalle encadrant son rayon.

4.3.4 Communication entre agents mobiles

Communication Peer to Peer

La gestion de la communication au sein de la société est centralisé, l'ordinateur central collecte les informations à partir de l'environnement, élabore les consignes et transmet à chaque robot les nouvelles consignes le concernant.

Le système de communication radio fréquence [20, 21], illustré dans la figure 4.19, est

intégré au système de contrôle. La technique de communication retenue est une communication *Peer to Peer*. La communication est établie entre le système de contrôle (serveur) et le robot (client).

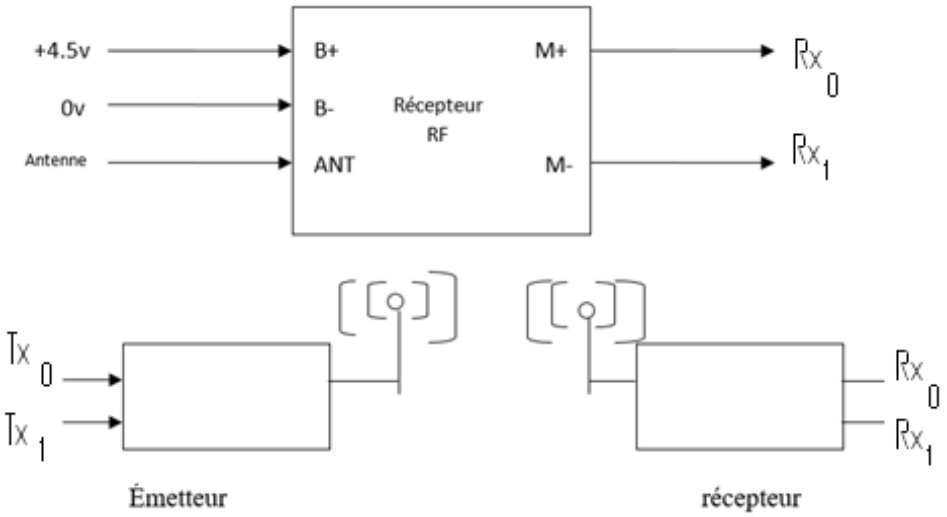


FIGURE 4.19 – Dispositif de communication sans fil

pour :

$$Tx_0 = 1 \rightarrow Rx_0 = 1 \tag{4.3}$$

$$Tx_1 = 1 \rightarrow Rx_1 = 1 \tag{4.4}$$

Avec la contrainte $Tx_0 \wedge Tx_1 \neq 1$, le réseau de Petri, illustré par la 4.20 est utilisé pour interpréter les commandes émises par le serveur et les traduire, au niveau du robot, en actions de déplacement et de changement de direction.

— Rx_0 : Déplacement du robot avant ou arrière.
— Rx_1 : Rotation du robot droite ou gauche.

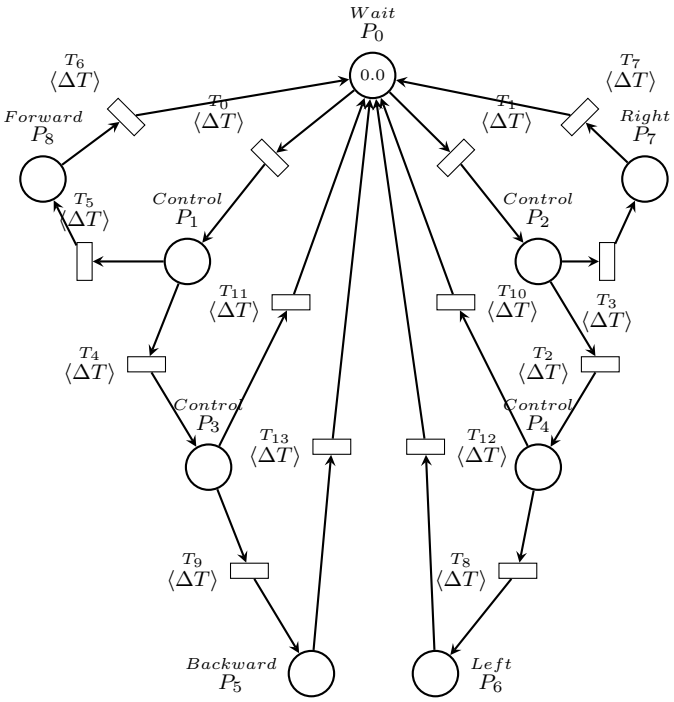


FIGURE 4.20 – Présentation de déplacement du rebot par le réseau de Petri

Avec : $T_0 = Rx_0.\overline{Rx_1}$ et $T_1 = \overline{Rx_0}.Rx_1$ et $T_2 = \overline{Rx_0}.\overline{Rx_1}$ et $T_3 = \overline{Rx_0}.Rx_1$ et $T_4 = \overline{Rx_0}.\overline{Rx_1}$ et $T_5 = Rx_0.\overline{Rx_1}$ et $T_6 = \overline{Rx_0}.\overline{Rx_1}$ et $T_7 = \overline{Rx_0}.\overline{Rx_1}$ and $T_8 = \overline{Rx_0}.Rx_1$ et $T_9 = Rx_0.\overline{Rx_1}$ et $T_{10} = \overline{Rx_0}.\overline{Rx_1}$ et $T_{11} = \overline{Rx_0}.\overline{Rx_1}$ et $T_{12} = \overline{Rx_0}.\overline{Rx_1}$ et $T_{13} = \overline{Rx_0}.\overline{Rx_1}$

Communication Broadcast

Le système de contrôle central, jouant le rôle de serveur, émet les informations en mode *Broadcast* et les robots, jouant le rôle de client, et se trouvant dans l'état d'écoute "Listen" reçoivent tous les messages émis. Mais seul le robot dont l'adresse est spécifiée clairement dans un champ du message, est autorisé à interpréter le message et à le traduire en commandes de déplacement et de rotation.

Le module utilisé pour l'émission est de type : **MX-FS-03V** et les modules de réception dont de type : **MX-05** . La fréquence de travail retenue est de 433 MHz (Figure 4.21).

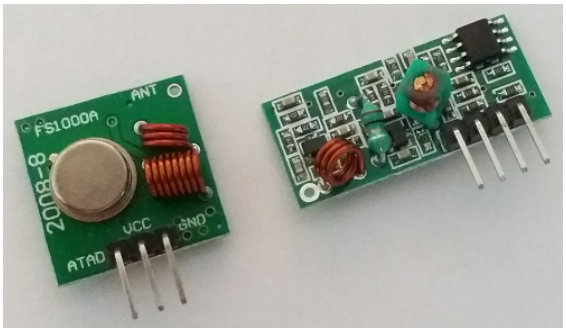


FIGURE 4.21 – Paire émetteur/récepteur radiofréquence

Le tableau 4.6 résume les caractéristiques technique des communications :

Module récepteur		**Module émetteur**	
Modèle	MX-05V	**Modèle**	MX-FS-03V
Alimentation	5V DC / 4mA	**Alimentation**	3.5-12 V
Fréquence de réception	433,92 MHz	**Fréquence d'émission**	433 MHz
Sensibilité de réception	-105 dB	**Distance de transmission**	20 – 200 m (selon la tension d'alimentation)
Dimensions	30 × 14 × 7 mm	**Dimensions**	19 × 19 mm
		Taux de transfert	4 ko/s
		Puissance	10 mW

TABLE 4.6 – Caractéristiques des modules d'émission/réception

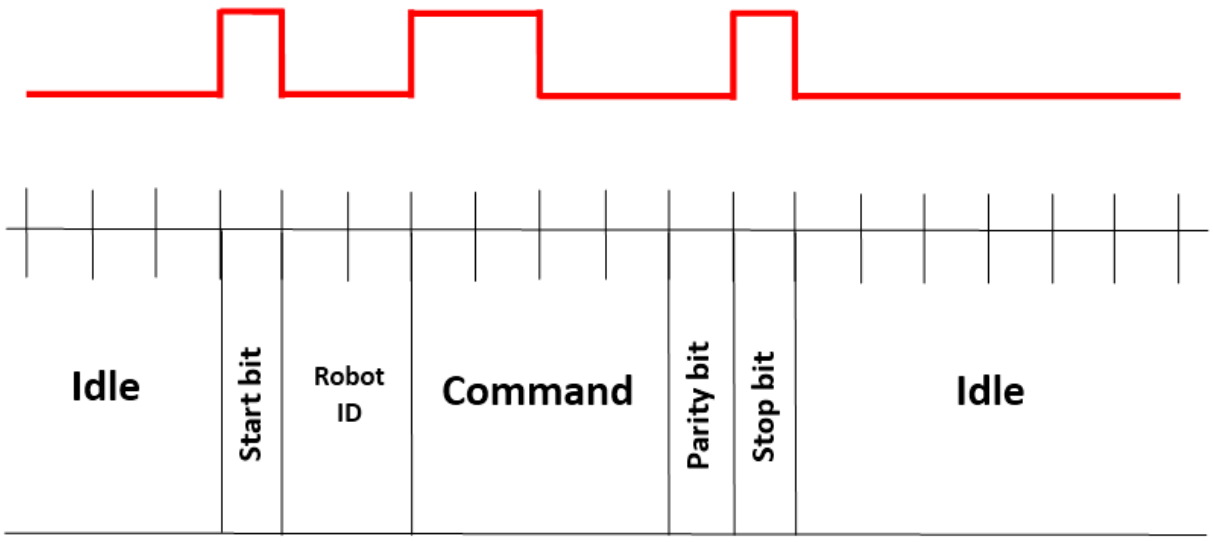


FIGURE 4.22 – Trame de communication sans fil

La trame de communication sans fil contient cinq champs :

— **Champ 1 :** un bit de démarrage (HIGH).
— **Champ 2 :** Deux bits de l'ID du robot, permettent d'adresser quatre robots. Ce champ peut être élargi pour adresser davantage de robots.
— **Champ 3 :** Quatre bits de commande, permettent de sélectionner l'une des quatre actions possible (Forward, Backward, Left et Right) pour faire évoluer un robot à deux roues différentielles.
— **Champ 4 :** Un bit de parité pour confirmer l'authenticité de la trame.
— **Champ 5 :** Un bit d'arrêt (HIGH) pour confirmer la fin de la commande.

4.3.5 Interface graphique : MATLAB

L'interface graphique de MATLAB, représenté dans la figure 4.23, offre un panel de commandes facilitant l'utilisation et la configuration de l'application pratique.

L'interface graphique comporte les éléments suivants :

— **Commande N°1 :** Retour d'état (position et angle) du Robot en utilisant le périphérique d'acquisition vidéo.
— **Commande N°2 :** Configuration de la partie acquisition de données ; détection du robot (grand cercle noir et de sa direction (petit cercle blanc).
— **Commande N°3 :** Choix de la méthode de recherche.
— **Commande N°4 :** Affichage des graphes de visibilité et de zones et le chemin de parcours par la méthode de navigation.
— **Commande N°5 :** Établir la connexion série.
— **Commande N°6 :** Navigation manuelle/automatique du robot.
— **Commande N°7 :** Messages importants (erreurs, position de robot, etc.)

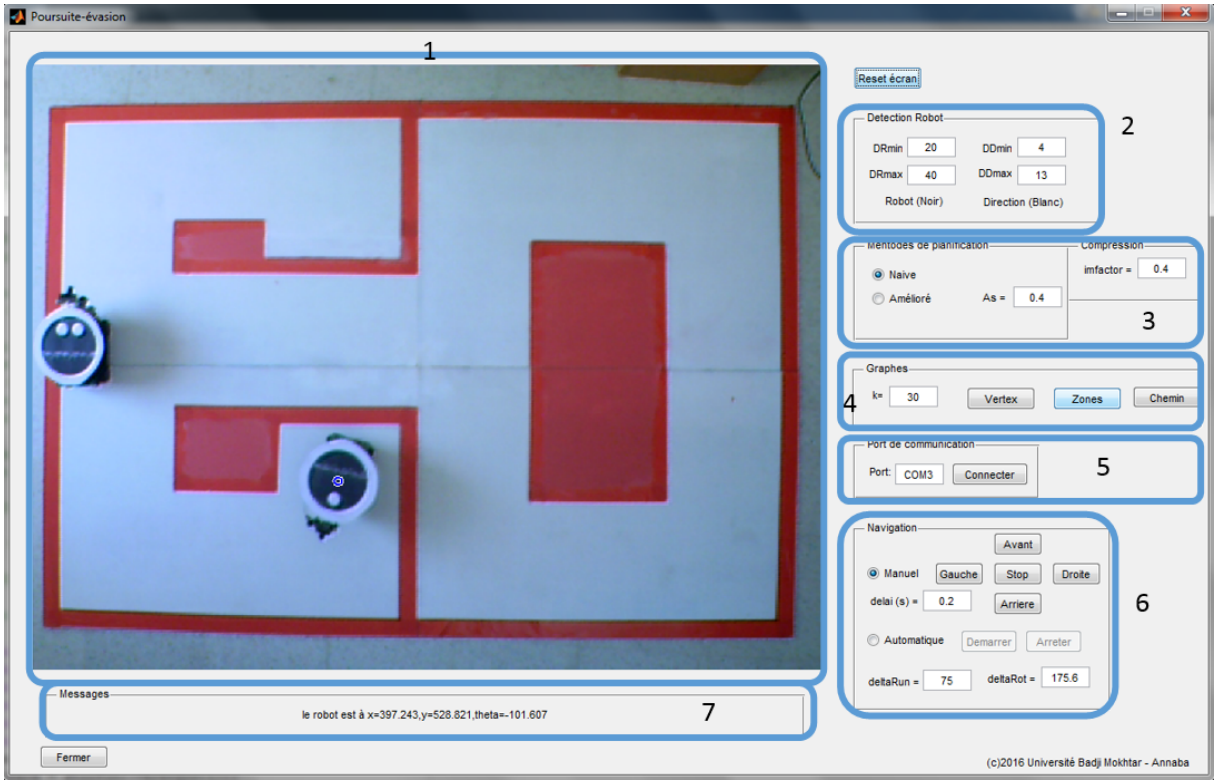
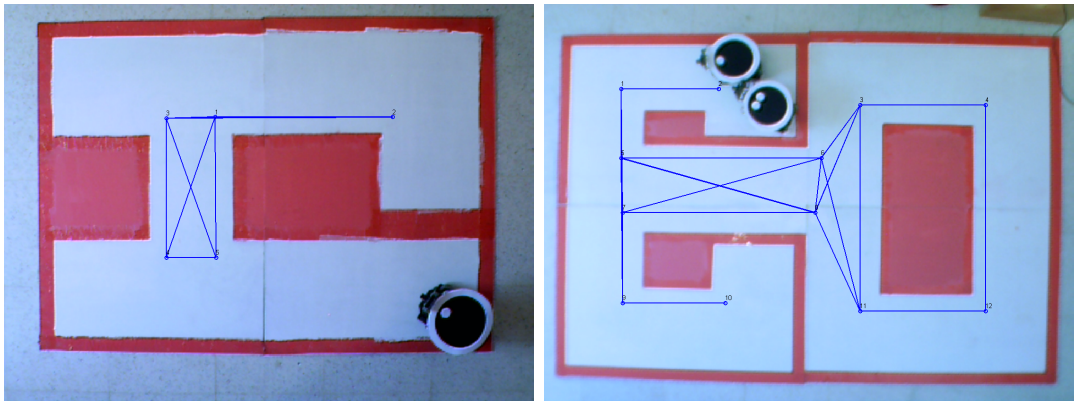


FIGURE 4.23 – Interface graphique de l'utilisateur

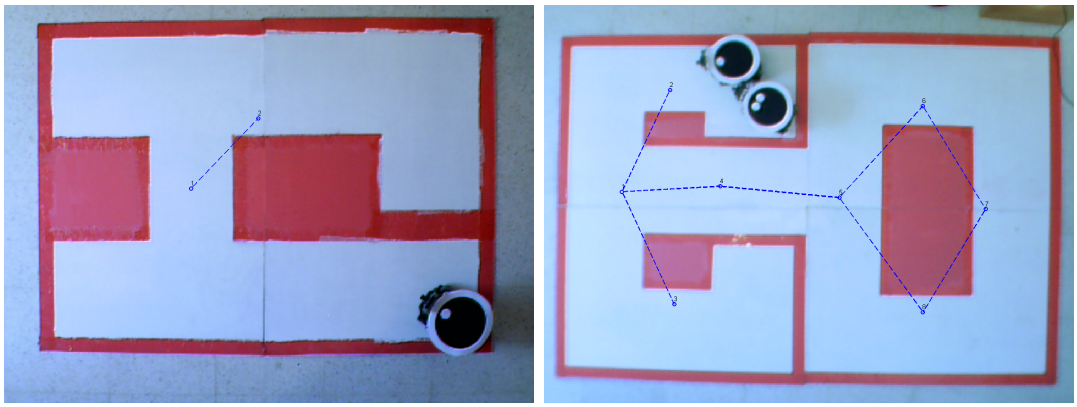
4.3.6 Modélisation de l'environnement en zones connexes

Sur les figures 4.24 et 4.25 sont représentés les graphes de visibilité et les graphes de zones des deux environnements tests. L'environnement (a) est constitué de deux zones reliées, il est d'une grande simplicité pour les opérations de décontamination, alors que l'environnement (b) présente huit (08) zones, dont deux sont critiques, et sa décontamination nécessite la collaboration de plusieurs agents robots.



(a) Environnement 1. (b) Environnement 2.

FIGURE 4.24 – Graphe de visibilité des environnements test.



(a) Environnement 1. (b) Environnement 2.

FIGURE 4.25 – Graphe de zones des environnements test.

4.3.7 Méthodes de décontamination

L'environnement 1 ne présente aucun intérêt pour l'opération de décontamination, il permet de mettre en valeur l'avantage de définir des trajectoires basées sur les techniques du potentiel. Par contre l'environnement 2, en contenant deux zones critiques, va nécessiter la mobilisation de trois agents pour le décontaminer par la méthode naïve et uniquement deux agents par la méthode améliorée.

Figure 4.26 présente les déplacements de l'agent gardien pour visiter tous les vertex en suivant un chemin optimisé des points de vue rapidité et sécurité.

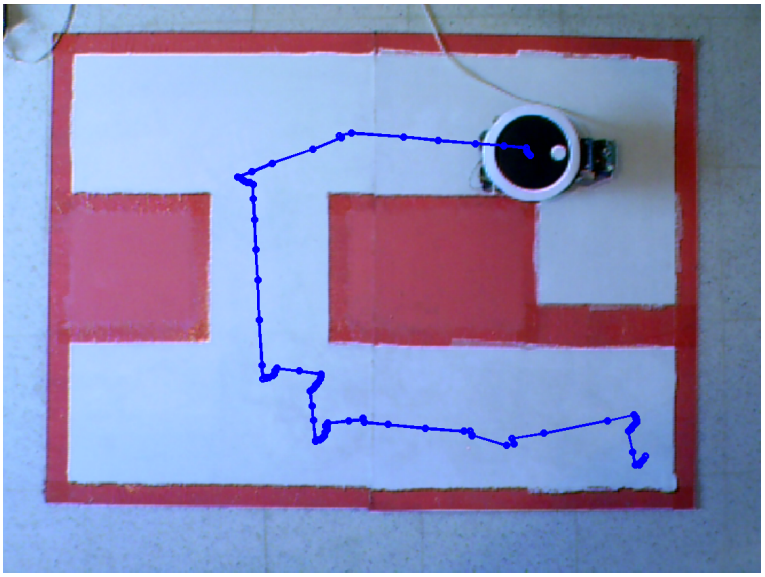


FIGURE 4.26 – Déplacement du robot 1 à l'environnement 1.

la figure 4.27 illustre la décontamination de l'environnement par la technique naïve. Du fait que seulement deux robots mobiles étaient disponibles pour l'application pratique,

l'agent gardien manquant est représenté par un disque noir. Ceci ne nuit absolument pas aux résultats obtenus.

La figure 4.28 illustre la décontamination par la technique améliorée où deux agents suffisent pour assurer la recherche complète de l'environnement.

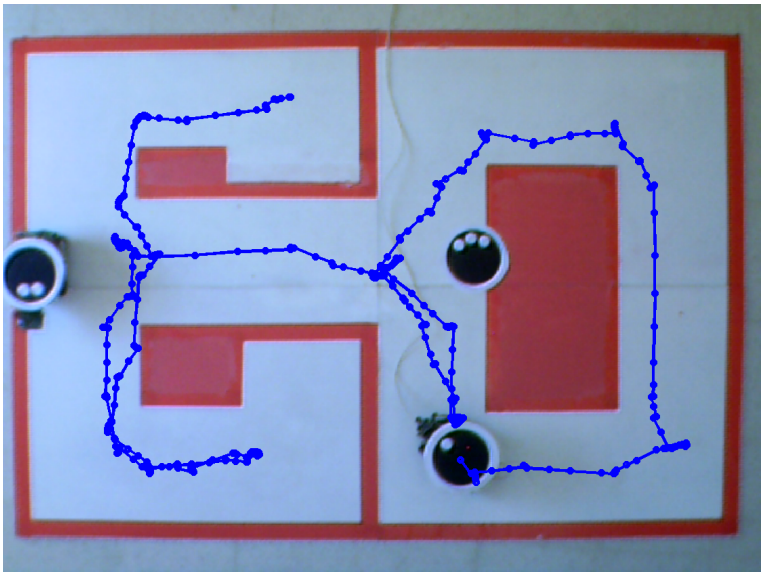


FIGURE 4.27 – Déplacement du robot 1 à l'environnement 2, méthode naïve.

4.3.8 Réseaux de Petri et validation

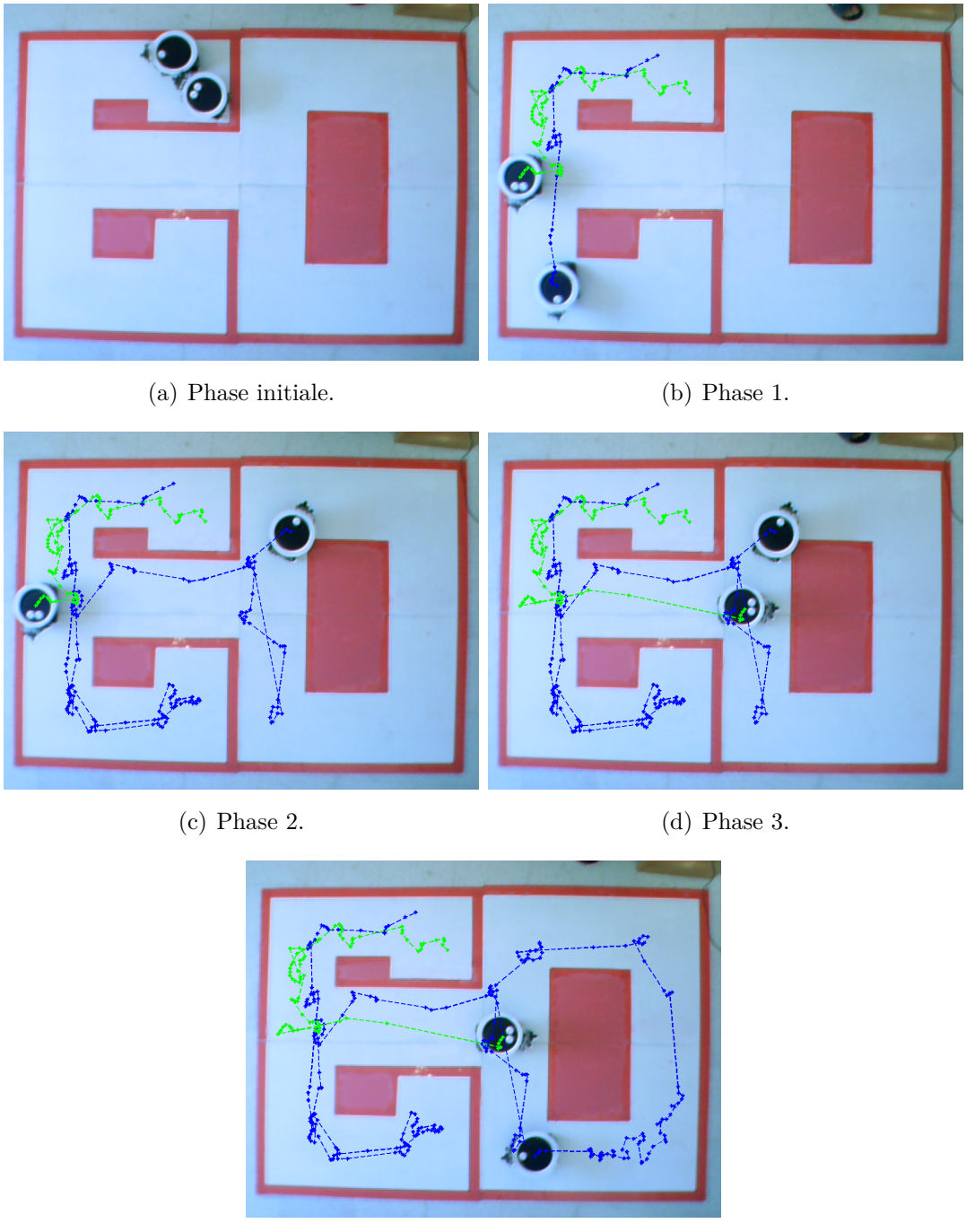
La figure A.3 dans la page 118 présente le réseau de Petri de l'environnement 2, l'équation 4.5 présente l'évolution du marquage pour la méthode naïve ainsi que l'équation 4.6 présente l'évolution du marquage pour la méthode améliorée.

On constate que, quel que soit la méthode de décontamination, le marquage final des places $contamA_i$? Est un vecteur nul.

$$M_0 = \begin{bmatrix} A_1 & contamA_1 \\ A_2 & contamA_2 \\ A_3 & contamA_3 \\ A_4 & contamA_4 \\ A_5 & contamA_5 \\ A_6 & contamA_6 \\ A_7 & contamA_7 \\ A_8 & contamA_8 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 3 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \xrightarrow{\sigma_1} \begin{bmatrix} 3 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \xrightarrow{\sigma_2} \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 2 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \xrightarrow{\sigma_3} \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} = M_{final} \qquad (4.5)$$

Avec :

— $\sigma_1 = 3 \times Forward_{21}$



(a) Phase initiale.

(b) Phase 1.

(c) Phase 2.

(d) Phase 3.

(e) Phase 4.

FIGURE 4.28 – Déplacement des robots 1 et 2 dans l'environnement 2 par la méthode améliorée.

— $\sigma_2 = 1 \times Forward_{13/31} \rightarrow 2 \times Forward_{14} \rightarrow 2 \times Forward_{45}$
— $\sigma_3 = 1 \times Forward_{56} \rightarrow 1 \times Forward_{67} \rightarrow 1 \times Forward_{78}$

$$M_0 = \begin{bmatrix} A_1 & contamA_1 \\ A_2 & contamA_2 \\ A_3 & contamA_3 \\ A_4 & contamA_4 \\ A_5 & contamA_5 \\ A_6 & contamA_6 \\ A_7 & contamA_7 \\ A_8 & contamA_8 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 2 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \overset{\sigma_1}{\rightarrow} \begin{bmatrix} 2 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \overset{\sigma_2}{\rightarrow} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 2 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \overset{\sigma_3}{\rightarrow} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} = M_{final} \qquad (4.6)$$

Avec :

— $\sigma_1 = 2 \times Forward_{21}$
— $\sigma_2 = 1 \times Forward_{13/31} \rightarrow 2 \times Forward_{14} \rightarrow 2 \times Forward_{45}$
— $\sigma_3 = 1 \times Forward_{56} \rightarrow 1 \times Forward_{67} \rightarrow 1 \times Forward_{78}$

4.4 Conclusion

L'implémentation des techniques de recherche et de planification de trajectoires, en simulation sur la plate-forme logicielle MobotSim, et leurs exécutions sur des environnements réels d'une part, et en pratique sur la plate forme réelle et leurs exécutions sur des environnements conçus en tenant compte des contraintes liées à l'éclairage, à la taille des robots et aux performances du système d'acquisition des images d'autre part, ont donné des résultats plus que satisfaisants. Quelque soit la complexité des environnements étudiés, chacune des méthodes de recherche a montré sa robustesse, en donnant des résultats, confirmés et validés par l'outil des réseaux de Petri.

Conclusion Générale

Ce travail de recherche traite des techniques de localisation, d'un intrus pour le neutraliser ou d'un rescapé pour lui apporter assistance, dans un environnement donné. La modélisation topographique des environnements sous forme de graphes de zones, nous a permis de développer des stratégies de décontamination, tout en optimisant le nombre d'agents mobiles nécessaires à la réussite de la mission. Et les réseaux de Petri ont été utilisés pour valider les techniques de résolution du problème de poursuite-évasion.

Les techniques de propagation de potentiel, inspirée de la méthode originale présentée par O. Khatib, permettent d'associer à chaque cellule de l'environnement disponible à la navigation un couple de valeurs de potentiel associé à la sécurité et à la rapidité des déplacements. A partir de ces cartes de potentiel, plusieurs techniques de génération de trajectoires ont été développées.

Pour des environnements connus, la méthode naïve est d'une grande efficacité, mais elle mobilise, de manière permanente, un nombre important de ressource en agents pour l'exploration de l'environnement. L'amélioration de la recherche, que nous avons développée, a permis d'optimiser, de manière dynamique, le nombre d'agents engagés.

Pour des environnements partiellement connus, où l'agent principal, en fonction de sa position courante et de sa perception de l'environnement, a le choix entre deux stratégies de progression dans l'exploration et la décontamination de l'environnement. La première stratégie, appelée stratégie *forward* donne de bons résultats, quelque soit la complexité de l'environnement, du point de vue temps de décontamination. Mais son principal problème réside dans la mobilisation d'agents de manière excessive pour protéger les zones qui n'ont pas besoin d'être gardées. La seconde technique, appelée stratégie *forward-backward*, basée sur l'idée de la visite de tous les sommets dans la zone actuelle, est très efficace pour décontaminer l'environnement, en mobilisant le nombre minimal d'agents gardien, mais elle souffre du problème de temps nécessaire pour explorer l'environnement dans sa totalité.

Les réseaux de Petri, utilisés comme outils de validation, ont confirmés l'efficacité et la robustesse des différentes techniques de recherche développées.

L'implémentation de la plate forme matérielle, malgré les difficultés et contraintes liées aux nombre réduit de robots et leur nature d'une part, et les limites des performances liées à la caméra utilisée induisant la conception d'environnements tests de taille modeste

et de surface plane d'autre part, a permis de valider de manière expérimentale, toutes les techniques de navigation et de décontamination développées dans ce travail. Cette partie pratique de notre travail, nous a amené à nous confronter aux problèmes d'acquisition d'images réelles et aux difficultés des communications radio fréquence.

Les perspectives ouvertes par ce travail sont nombreuses et certaines sont d'une grande actualité dans le domaine de la gestion des catastrophes naturelles. Il peut être poursuivi en étendant son étude aux environnements dynamiques (ouverture ou fermeture de portes entraînant le changement du modèle topologique de l'environnement), de valider les techniques de navigation et de décontamination, développées dans ce travail, pour des environnements en 3 D (environnements proches de la réalité).

Il peut être poursuivi, en le complétant, par le développement d'une plate forme logicielle interactive de modélisation et de validation, de toutes les techniques étudiées et présentées dans ce travail.

Bibliographie

[1] Mazda Ahmadi and Peter Stone. Keeping in touch : Maintaining biconnected structure by homogeneous robots. In *the Association for the Advancement of Artificial Intelligence*, 2006.

[2] Marsan Marco Ajmone. *Modelling with Generalized Stochastic Petri Nets*. John Wiley & Sons inc., 1994.

[3] Christos Ampatzis, Elio Tuci, Vito Trianni, Anders Lyhne Christensen, and Marco Dorigo. Evolving self-assembly in autonomous homogeneous robots : Experiments with two physical robots. *Artificial Life*, 15(4), 2009.

[4] Isaac Asimov. *Runaround*. Street and Smith Publications, 1942.

[5] Tamer Basar and Geert Jan Olsder. *Dynamic Noncooperative Game Theory, 2nd edition*. Society for Industrial and Applied Mathematics, 1999.

[6] Bernard Berthomieu and Michel Diaz. Modeling and verification of time dependent systems using time petri nets. *IEEE Trans. on Software Eng.*, 17(3) :259–273, 1991.

[7] Andrew Beveridge and Yiqing Cai. Two-dimensional pursuit-evasion in a compact domain with piecewise analytic boundary. *arXiv preprint arXiv :1505.00297*, 2015.

[8] Pere Bonet, Catalina M. Llado, and Ramon Puigjaner. Pipe v2.5 : a petri net tool for performance modeling. In *the 23rd Latin American Conference on Informatics (CLEI2007)*, 2007.

[9] Rodney A. Brooks and Anita M. Flynn. Fast, cheap and out of control. Technical report, MIT AI Laboratory, 1989.

[10] Jean-Paul Calvez. *Spécification et conception des systèmes*. Masson, 1990.

[11] Hervé Camus. Conduite de systèmes flexibles de production manufacturière par composition de régimes permanents cycliques : modélisation et évalution de performances à l'aide des réseaux de petri. Technical report, Université de Lille 1, 1997.

[12] Rajib Kumar Chatterjee, Neha Neha, and Anirban Sarkar. Behavioral modeling of multi agent system : High level petri net based approach. *International Journal of Agent Technologies and Systems (IJATS)*, 7(1) :55–78, January 2015.

[13] Vasek Chvatal. A combinatorial theorem in plane geometry. *Journal of Computorial Theory (B)*, 1975.

[14] Alessio Conte. Review of the bron-kerbosch algorithm and variations. Technical report, University of Glasgow, 2013.

[15] Hannah Daugherty. Decoration ideas : Office building floorplans, 2013.

[16] Reinhard Diestel. *Graph Theory, Third Edition*. Springer-Verlag, 2005.

[17] Adel Djellal and Rabah Lakel. Using graph search technique to solve the pursuit-evasion problem in unknown environment. In *International Conference On Industrial Engineering and Manufacturing ICIEM'10*, 2010.

[18] Adel Djellal and Rabah Lakel. Using graph theory to search an evader in unknown environments. In *Conférence Internationale sur le Traitement de l'Information Multimédia (CITIM'2012)*, 2012.

[19] Adel Djellal, Rabah Lakel, and Randa Chergui. Navigating mobile robot in finite environment using potential field merging. *International Journal of Control and Automation*, 2015.

[20] Adel Djellal, Rabah Lakel, and Brahim Lakehal. Radio frequency control of a robot to find optimal-safe path in finite environment. In *International Conference on Embedded Systems in Telecommunications and Instrumentation (ICESTI'14)*, 2014.

[21] Adel Djellal, Rabah Lakel, Brahim Lakehal, and Ines Belazzoug. Controlled robot with rf system in finite known environment. In *International Congress on Telecommunication and Application'14 (ICTA'14)*, 2014.

[22] ezblueprint.com. A functional office with office cubicals, reception, conference rooms, 2015.

[23] Alessandro Farinelli, Luca Iocchi, Daniele Nardi, and Vittorio Amos Ziparo. Assignment of dynamically perceived tasks by token passing in multirobot systems. In *IEEE*, volume 94, pages 1271–1288, Jully 2006.

[24] Japan Science Foundation. Science museum, 2008.

[25] Herbert Freeman. On the encoding of arbitrary geometric configurations. *IRE Transactions on Electronic Computers*, 1961.

[26] from Animals to Animats14. the 14th international conference on the simulation of adaptive behavior. In *SAB2016*, 2016.

[27] Leonidas Guibas, Jean-Claude Latombe, Steven M. LaValle, David Lin, and Rajeev Motwani. A visibility-based pursuit-evasion problem. *International Journal of Computational Geometry and Applications,*, 1985.

[28] Rémy Guyonneau. *Méthodes ensemblistes pour la localisation en robotique mobile*. PhD thesis, Université d'Angers, 2013.

[29] Joao P. Hespanha, Hyoun Jin Kim, and Shankar Sastry. Multiple-agent probabilistic pursuit-evasion games. In *38th IEEE Conf. on Decision and Control*, 1999.

[30] Joao P. Hespanha, Maria Prandini, and Shankar Sastry. Probabilistic pursuit-evasion games : a one-step nash approach. In *39th IEEE Conf. on Decision and Control*, 2000.

[31] American Building Calculations Inc. Parkside plaza one, 2005.

[32] Rufus Isaacs. *Differential Games i.* Wiley, 1954.

[33] Wael Khansa. *Reseaux de Petri p-Temporels Contribution a l'Etude des Systemes a evenements Discrets*. PhD thesis, L'Univesite de Savoie, 1997.

[34] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotic Research*, 1986.

[35] K. Krishnamoorthy, D. Casbeer, and M. Pachter. Pursuit on a graph using partial information. In *American Control Conference (ACC), IEEE*, 2015.

[36] Rabah Lakel and Adel Djellal. Resolving the pursuit evasion problem in known environment using graph theory. *Int. J. Bio-Inspired Computation*, 2(6) :434–439, 2010.

[37] Steven M. LaValle and John E. Hinrichsen. Visibility-based pursuit-evasion : the case of curved environments. In *IEEE Int. Conf. Robot. & Autom.*, 1999.

[38] Steven M. LaValle, David Lin, Leonidaa J. Guibas, Jean-Claude Latombe, and Rajeev Motwani. Finding an unpredicable target in a workspace with obstacles. In *IEEE Int. Conf. Robot. & Autom.*, 1997.

[39] Studio Libeskind. Denver art museum, 2010.

[40] Anna Lubiw, Jack Snoeyink, and Hamideh Vosoughpour. Visibility graphs, dismantlability, and the cops and robbers game. *arXiv preprint arXiv :1601.01298*, 2016.

[41] P. Merlin. *A study of the recoverability of communication protocols*. PhD thesis, University of California, 1974.

[42] MobotSoft. Wall following robot "mobotsoft". Technical report, WordPress, 2016.

[43] Tadao Murata. Petri nets, properties, analysis and applications. In *the IEEE*, 1989.

[44] Naoki Nishikawa, Reiji Suzuki, and Takaya Arita. Coordination control design of heterogeneous swarm robots by means of task-oriented optimization. *Artificial Life and Robotics*, 2016.

[45] Eugenio Oliveira, Klaus Fischer, and Olga Stepankova. Multi-agent systems : which research for which applications. *Robotics and Autonomous Systems*, 1999.

[46] openbook.info. Plan de bureaux commerciaux, 2016.

[47] Perimeter Park. Floor plan for nashville office space at perimeter park executive center, 2012.

[48] POB-Technology. Pob-bot user's manual. Technical report, POB-Technology, 2008.

[49] Jean-Marie Proth and Xiaolan Xie. *Les Réseaux de Petri pour la Conception et la Gestion des Systèmes de Production*. Masson, 1994.

[50] Christiaan Scheepers and Andries P. Engelbrecht. Training multi-agent teams from zero knowledge with the competitive coevolutionary team-based particle swarm optimiser. *Soft Computing*, 2016.

[51] Reid Simmons, David Apfelbaum, Dieter Fox, Robert P. Goldman, Karen Zita Haigh, David J. Musliner, Michael Pelican, and Sebastian Thrun. Coordinated deployment of multiple, heterogeneous robots. In *IEEE/RSJ International Conference on. Vol. 3. IEEE*, 2000.

[52] Alexander Spröwitz, Rico Moeckel, Massimo Vespignani, Stephane Bonardi, and Auke Jan Ijspeert. Roombots : A hardware perspective on 3d self-reconfiguration and locomotion with a homogeneous modular robot. *Robotics and Autonomous Systems*, 62(7) :1016–1033, July 2014.

[53] Paolo Stegagno, Marco Cognetti, Lorenzo Rosa, Pietro Peliti, and Giuseppe Oriolo. Relative localization and identification in a heterogeneous multi-robot system. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[54] Larry M. Stephens and Matthias B. Merx. The effect of agent control strategy on the performance of a dai pursuit problem. In *the 1990 Distributed AI Workshop*, 1990.

[55] Volker Stix. Finding all maximal cliques in dynamic graphs. *Computational Optimization and Applications*, 7(2), 2004.

[56] The Institute For American Indian Studies. Floor plans & grounds, 2016.

[57] 120 Market Suites. Etage 7, 2016.

[58] Ichiro Suzuki and Masafumi Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM J. Computing*, 21(5) :863–888, October 1992.

[59] René Vidal, Shahid Rashid, Cory Sharp, Omid Shakernia, Jin Kim, and Shankar Sastry. Pursuit-evasion games with unmanned gound and aerial vehicles. In *IEEE Int. Conf. Robot. & Autom.*, 2001.

[60] René Vidal, Omid Shakernia, H. Jin Kim, David Hyunchul Shim, and Shankar Sastry. Probabilistic pursuit-evasion games : Theory, implementation and experimental evaluation. *IEEE Transactions on Robotics and Automation*, 2002.

[61] Kai M. Wurm, Christian Dornhege, Bernhard Nebel, Wolfram Burgard, and Cyrill Stachniss. Coordinating heterogeneous teams of robots using temporal symbolic planning. *Autonomous Robots*, 2013.

[62] Louchka Popova Zeugmann. *Time and Petri Nets*. Springer Heidelberg, 2013.

[63] Vittorio A. Ziparo, Luca Iocchi, Daniele Nardi, Pier Francesco Palamara, and Hugo Costelha. Pnp : A formal model for representation and execution of multi-robot plans. In *7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, 2008.

[64] Vittorio Amos Ziparo and Luca Iocchi. Petri net plans. In *Fourth International Workshop on Modelling of Objects, Components, and Agents (MOCA'06)*, 2006.

Annexe A

Réseaux de Petri

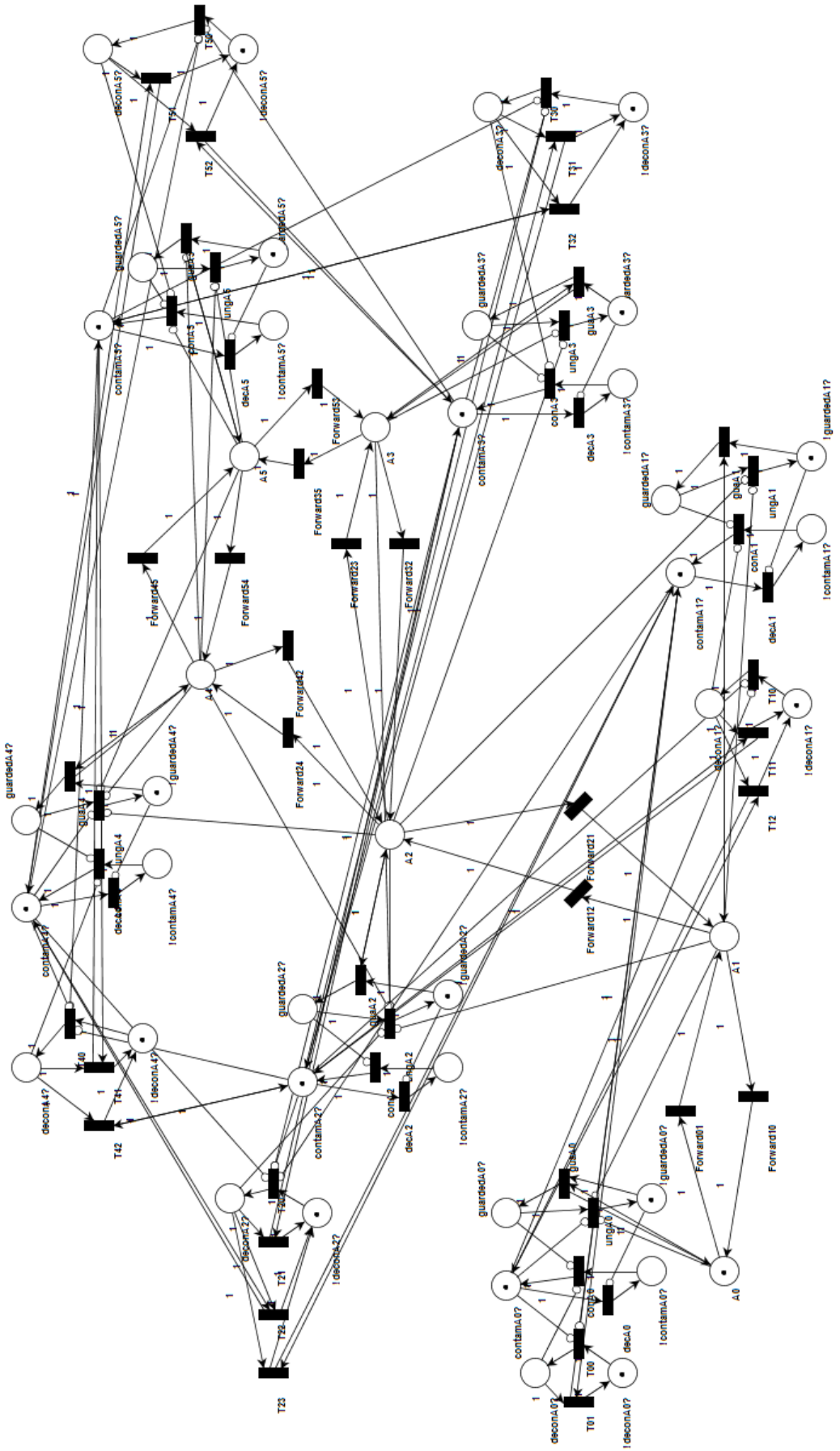


FIGURE A.1 – Exemple d'environnement modélisé par les réseaux de Petri

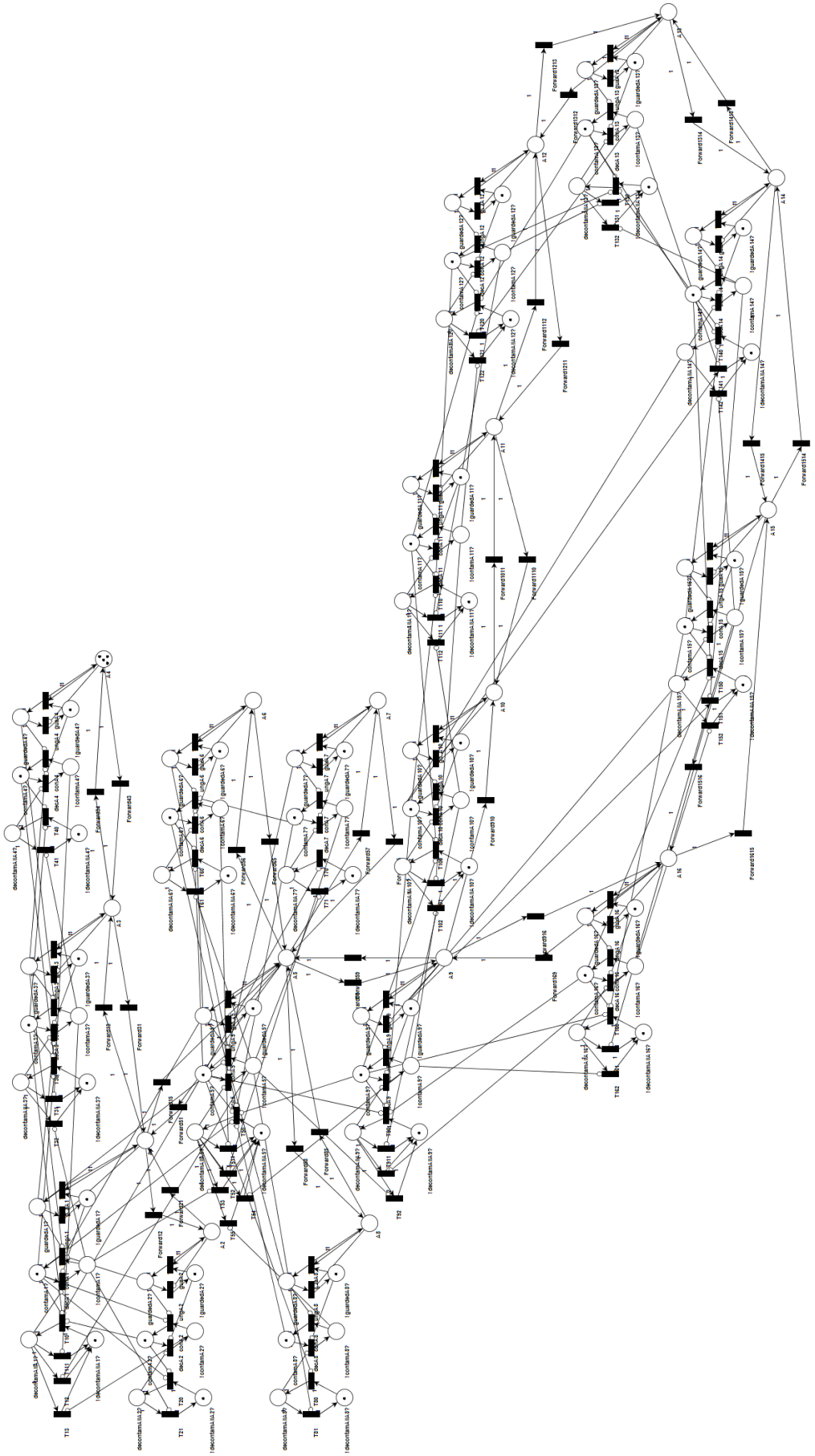


FIGURE A.2 – Réseau de Petri de l'environnement E.

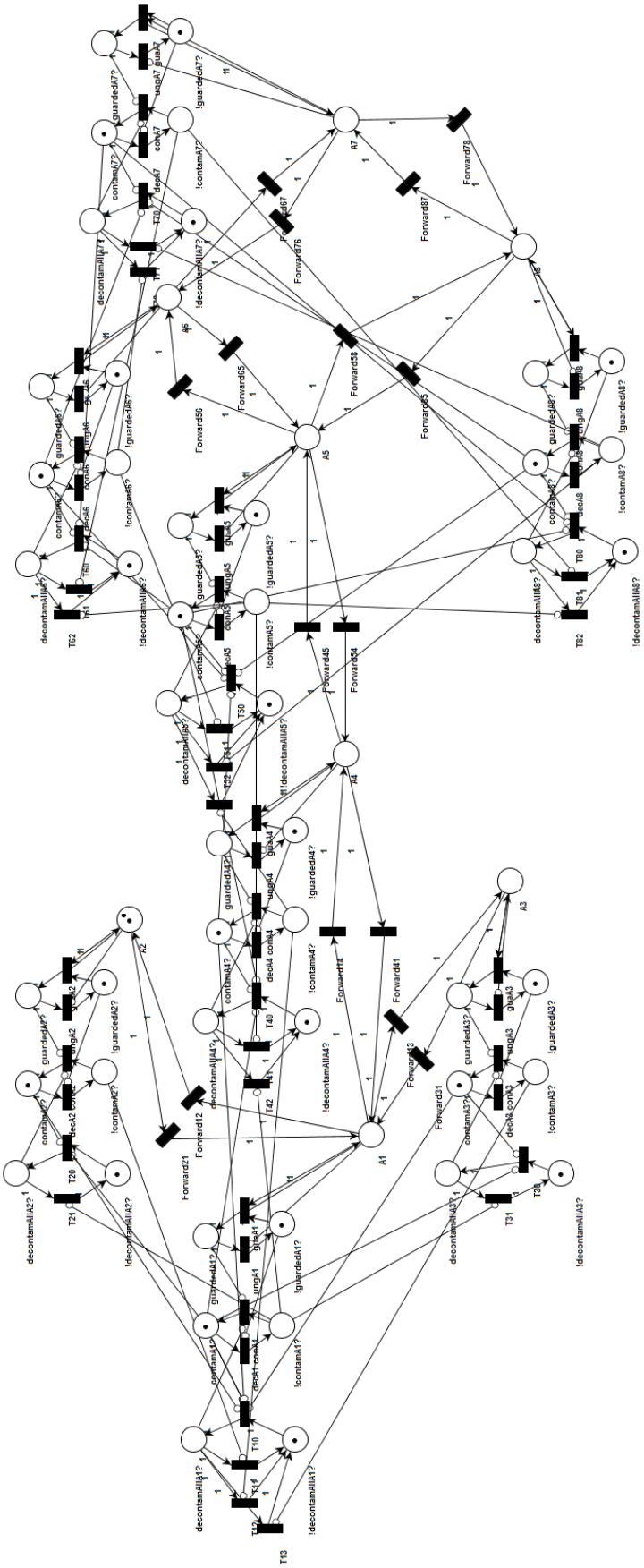


FIGURE A.3 – Réseau de Petri de l'environnement 2 des tests réels.